

**A Thesis Submitted for the Degree of PhD at the University of Warwick**

**Permanent WRAP URL:**

<http://wrap.warwick.ac.uk/159992>

**Copyright and reuse:**

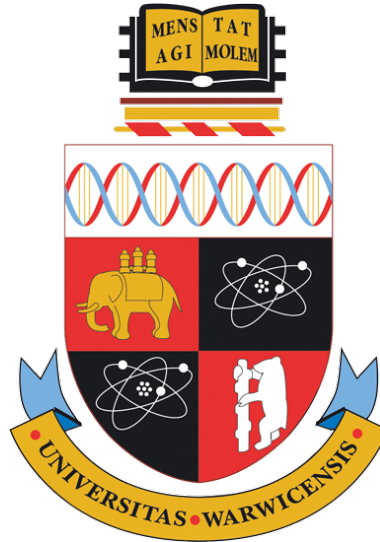
This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: [wrap@warwick.ac.uk](mailto:wrap@warwick.ac.uk)



# Reducing Spatio-Temporal Data: Methods and Analysis

by

**Liam Steadman**

**Thesis**

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

**Doctor of Philosophy**

**Department of Computer Science**

December 2020

# Contents

<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>Acknowledgments</b>	<b>ix</b>
<b>Declarations</b>	<b>x</b>
<b>Abstract</b>	<b>xi</b>
<b>Acronyms</b>	<b>xiv</b>
<b>Notation</b>	<b>xv</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 Objectives . . . . .	4
1.2 Contributions . . . . .	5
<b>Chapter 2 Background</b>	<b>7</b>
2.1 Spatio-Temporal Analysis Tasks . . . . .	7
2.1.1 Analysing and Learning from Spatio-Temporal Data . .	7
2.1.2 Linking Spatio-Temporal Datasets . . . . .	9
2.2 Reducing Spatio-Temporal Datasets . . . . .	11
2.2.1 Feature Selection Techniques . . . . .	11
2.2.2 Feature Extraction Techniques . . . . .	12
2.2.3 Instance Selection and Abstraction Techniques . . . . .	13
2.2.4 Data Sketching Techniques . . . . .	14
2.2.5 Reducing using Modelling Techniques . . . . .	15
2.3 Partitioning Datasets . . . . .	17
2.4 Modelling Spatio-Temporal Data . . . . .	18
2.5 Evaluating Spatio-Temporal Methods . . . . .	20
2.5.1 Errors in Location of Sensors . . . . .	20
2.5.2 Errors in Timestamps of Instances . . . . .	21
2.5.3 Noise in Feature Values . . . . .	22

2.5.4	Sources of Missing Instances . . . . .	23
2.6	Discussion . . . . .	24
<b>Chapter 3</b>	<b>Datasets</b>	<b>26</b>
3.1	Characterising Spatio-Temporal Data . . . . .	27
3.1.1	Coverage of Space and Time . . . . .	28
3.1.2	Feature Values . . . . .	31
3.2	Datasets Used in this Thesis . . . . .	33
3.2.1	Road Traffic . . . . .	33
3.2.2	Air Temperature . . . . .	38
3.2.3	Rainfall . . . . .	40
<b>Chapter 4</b>	<b>Reducing by Partitioning and Modelling</b>	<b>44</b>
4.1	Reducing Datasets to Partitions and Models . . . . .	45
4.1.1	Partitioning and Modelling Spatio-Temporal Data . . .	45
4.1.2	Evaluating Reduced Datasets . . . . .	48
4.2	$k$ D-STR: $k$ -Dimensional Spatio-Temporal Reduction Algorithm	51
4.2.1	Partitioning Technique of $k$ D-STR . . . . .	53
4.2.2	Modelling Techniques for $k$ D-STR . . . . .	56
4.2.3	Data Reduction Algorithm . . . . .	57
4.2.4	Data Reconstruction . . . . .	58
4.2.5	Analysis of Complexity . . . . .	60
4.3	Experimental Evaluation Methodology . . . . .	61
4.4	Results . . . . .	64
4.4.1	Trade-Off Between Error and Storage . . . . .	64
4.4.2	Choice of Modelling Technique . . . . .	67
4.4.3	Spatial and Temporal Properties Discovered . . . . .	71
4.4.4	Comparison with Existing Techniques . . . . .	72
4.4.5	Choice of Spatial Referencing System . . . . .	74
4.4.6	Impact of Input Dataset Size . . . . .	75
4.4.7	Running Time . . . . .	77
4.5	Discussion . . . . .	77
4.6	Summary . . . . .	80
<b>Chapter 5</b>	<b>Augmenting and Reducing Spatio-Temporal Data</b>	<b>81</b>
5.1	Augmenting Primary Datasets with Supplementary Datasets .	82
5.1.1	Methods for Imputing Supplementary Instances . . . . .	83
5.1.2	Choosing Parameters for Imputation Methods . . . . .	86
5.2	Reducing Supplementary Datasets . . . . .	87
5.2.1	Sampling the Primary Dataset . . . . .	89
5.2.2	Preprocessing for Further Reduction . . . . .	89
5.3	Integrating the Augmenting and Reduction Processes . . . . .	90



5.4	Experimental Evaluation Methodology . . . . .	92
5.4.1	Preprocessed Datasets . . . . .	94
5.4.2	Linking Procedure and Baseline Augmented Datasets . . . . .	95
5.5	Results . . . . .	97
5.5.1	Speedup Achieved Using Reduced Datasets . . . . .	98
5.5.2	Error in Linked Features . . . . .	99
5.5.3	Retention of Information in Space and Time . . . . .	101
5.5.4	Error in Original Supplementary Features . . . . .	103
5.6	Discussion . . . . .	103
5.7	Summary . . . . .	105

## **Chapter 6 Implications of Real-World Applications on Spatio-Temporal Reduction 107**

6.1	Mutual Reduction . . . . .	109
6.1.1	Mutual Partitioning Technique for $k$ D-STR . . . . .	111
6.1.2	Experimental Evaluation Methodology . . . . .	116
6.1.3	Results . . . . .	117
6.1.4	Discussion . . . . .	122
6.2	$Dk$ D-STR: Distributed Spatio-Temporal Reduction . . . . .	123
6.2.1	Adapting $k$ D-STR for Distributed Environments . . . . .	124
6.2.2	Experimental Evaluation Methodology . . . . .	131
6.2.3	Results . . . . .	132
6.2.4	Discussion . . . . .	134
6.3	Impact of Errors and Missing Instances in Input Datasets . . . . .	136
6.3.1	Sources of Error and Missing Data . . . . .	137
6.3.2	Experimental Evaluation Methodology . . . . .	141
6.3.3	Results . . . . .	143
6.3.4	Discussion . . . . .	150
6.4	Summary . . . . .	153

## **Chapter 7 Discussion 154**

7.1	Contributions of this Research . . . . .	154
7.1.1	Contribution 1: $k$ -Dimensional Spatio-Temporal Reduction . . . . .	154
7.1.2	Contribution 2: Augmenting and Reducing Spatio-Temporal Datasets . . . . .	155
7.1.3	Contribution 3: Mutually Reducing Datasets . . . . .	156
7.1.4	Contribution 4: Distributed Data Reduction . . . . .	156
7.1.5	Contribution 5: Robustness to Error . . . . .	157
7.2	Utility of Spatio-Temporal Reduction . . . . .	158
7.3	Limitations to Generalisation . . . . .	159

<b>Chapter 8 Conclusions and Future Work</b>	<b>161</b>
8.1 Future Work . . . . .	162
<b>Appendix A Supplementary Results for Chapter 4</b>	<b>165</b>
A.1 Trade-Off Between Error and Storage . . . . .	165
A.2 Reconstructed Sensor Values . . . . .	165
A.3 Volumes of Partitions . . . . .	169
A.4 Effect of Distance Metric . . . . .	170
A.5 Choice of Spatial Referencing System . . . . .	172
A.6 Impact of Input Dataset Size . . . . .	172
<b>Appendix B Creating Baseline Augmented Datasets</b>	<b>177</b>
<b>Appendix C Supplementary Results for Chapter 5</b>	<b>182</b>
C.1 Speedup Achieved Using Reduced Datasets . . . . .	182
C.2 Error in Linked Features . . . . .	184
C.3 Retention of Information in Space and Time . . . . .	186
C.4 Error in Original Supplementary Features . . . . .	188
<b>Appendix D Supplementary Results for Section 6.1</b>	<b>189</b>
D.1 Effect on Storage Used . . . . .	189
D.2 Effect on Error Incurred . . . . .	190
<b>Appendix E Survey of Distributed Hierarchical Clustering Techniques</b>	<b>192</b>
<b>Appendix F Supplementary Results for Section 6.2</b>	<b>194</b>
<b>Appendix G Selecting Data Samples for Perturbations</b>	<b>196</b>
<b>Appendix H Supplementary Results for Section 6.3</b>	<b>200</b>
H.1 Effect on Variance . . . . .	200
H.2 Effect on Partitioning . . . . .	200
H.3 Effect on Reconstruction Error and Storage . . . . .	207
H.4 Effect on Imputation Error . . . . .	212

# List of Tables

3.1	Characteristics of the datasets used in this thesis. . . . .	34
3.2	Percentage of instances present in the traffic datasets . . . . .	35
3.3	Number of sensors in traffic, air temperature and rainfall datasets	35
3.4	Percentage of instances present in air temperature and rainfall datasets . . . . .	41
4.1	Example footfall data recorded at the 9 sensors (A-K) shown in Figure 4.1 . . . . .	53
4.2	Mean number of partitions output by $k$ D-STR . . . . .	69
4.3	Comparison of storage used by existing techniques . . . . .	73
4.4	Mean number of partitions output as data volume increases . .	78
6.1	Storage used by the indepentently and mutually reduced datasets	119
6.2	Maximum number of instances on any worker node, averaged over all data samples for each dataset . . . . .	133

# List of Figures

3.1	Nearest neighbour imputation: 1D vs. 2D SRS . . . . .	29
3.2	Locations of sensors in traffic, air temperature and rainfall datasets	36
3.3	Distances between sensors and periods of inactivity for traffic, air temperature and rainfall datasets . . . . .	37
3.4	Total vehicle count, average speed and distribution of vehicles sizes for July 2017 . . . . .	39
3.5	Variograms of the traffic, air temperature and rainfall datasets	42
3.6	Distribution of feature values for traffic, air temperature and rainfall datasets . . . . .	43
4.1	Partitioning of instances in space and time . . . . .	47
4.2	Overview of the $k$ D-STR algorithm . . . . .	52
4.3	Relationship between clusters and partitions in $k$ D-STR . . . .	54
4.4	Partitions formed during the data partitioning process . . . .	55
4.5	NRMSE incurred and storage used by the reduced traffic, air temperature and rainfall datasets. . . . .	66
4.6	Histogram of number of coefficients output per partition . . . .	70
4.7	Comparison between $k$ D-STR (with DCT-P modelling) and PCA, IDEALEM and DEFLATE . . . . .	72
4.8	Comparison of NRMSE, storage and partitions output by $k$ D- STR for 1D vs. 2D SRS . . . . .	76
4.9	Effect of increasing data volume in temporal domain on NRMSE and storage used . . . . .	77
4.10	Effect of increasing data volume in spatial domain on NRMSE and storage used . . . . .	78
5.1	Empirical variograms of features from the supplementary datasets	96
5.2	Time taken to augment the primary dataset compared to kNN baseline . . . . .	98
5.3	Comparison of linked feature histograms created using raw data versus reduced data . . . . .	100

5.4	Comparison of error incurred by engineering features using the reduced datasets . . . . .	100
5.5	Comparison of error incurred by RES $k$ D-STR heuristic vs. LES heuristic . . . . .	101
5.6	Temporal locations of partitions with more than one model coefficient for the air temperature dataset . . . . .	102
5.7	Storage used and error incurred in original data features using RES heuristic vs. LES heuristic . . . . .	104
5.8	Reconstruction error in mean temperature feature for original supplementary dataset . . . . .	104
6.1	Initial partitioning of two datasets by AdaptDB . . . . .	110
6.2	Partitioning of two datasets by $Mk$ D-STR . . . . .	112
6.3	Overview of the $Mk$ D-STR algorithm . . . . .	113
6.4	Number of models output by $Mk$ D-STR versus $k$ D-STR . . . . .	120
6.5	Legend used for Figure 6.6 . . . . .	120
6.6	NRMSE incurred by $k$ D-STR versus $Mk$ D-STR . . . . .	121
6.7	Issue with preventing cross-boundary communication in distributed reduction scenarios . . . . .	124
6.8	Overview of the $Dk$ D-STR process . . . . .	130
6.9	Time taken and number of partitions output when reducing the datasets presented in Chapter 3 using $Dk$ D-STR . . . . .	135
6.10	Imputation error incurred by $Dk$ D-STR . . . . .	136
6.11	Number of partitions in the reduced form of the datasets with noise and spatial/temporal error . . . . .	146
6.12	Number of partitions in the reduced form of the datasets with missing instances . . . . .	147
6.13	Imputation error of the erroneous A30 dataset . . . . .	151

## List of Algorithms

1	$k$ D-STR algorithm for reducing spatio-temporal data . . . . .	59
2	Augmenting primary dataset $D^P$ with information from multiple supplementary datasets . . . . .	83

3	Reducing the supplementary datasets $D^1, \dots, D^n$ with $k$ D-STR and the LES heuristic . . . . .	91
4	Mutual $k$ D-STR (MkD-STR) algorithm for reducing multiple datasets simultaneously . . . . .	114
5	<i>incrCoefficients()</i> function for MkD-STR Algorithm (Algorithm 4)	115
6	<i>incrPartitions()</i> function for MkD-STR Algorithm (Algorithm 4)	115
7	Augmenting the road condition dataset with features calculated from the traffic, air temperature and rainfall datasets . . . . .	177

# Acknowledgments

First and foremost I would like to thank my supervisors Nathan Griffiths and Stephen Jarvis for their direction and encouragement over the past four years. Their guidance and mentorship has been phenomenal and this thesis would not have been possible without their help. I would also like to thank Caroline, Helen, Mark, Shaun and Stuart at TRL — their knowledge and support has been invaluable.

Second, I would like to thank Roger Packwood and the technical administration team in the Department of Computer Science for their assistance and flexibility while gathering the results in this thesis.

Third, I would like to thank the friends that have supported and emboldened me throughout my PhD. My sincerest gratitude goes to David and Helen, whose suggestions over tea have been limitless, and Katherine, Melissa and Matt Q, whose friendship has helped make the past four years more enjoyable than I could have imagined. I would also like to thank John and Matt B, and those from WISC and DCS who have helped and entertained in countless ways throughout my PhD.

Finally, I would like to offer my deepest gratitude to my family. To Nat, whose encouragement and affection has helped me persevere through the most demanding of days. To my grandparents, Margaret and Jim, Jean and Keith, whose support is never-ending. And my mum, Vicky, who has championed me every day, and without whom this thesis could not exist.

# Declarations

Parts of this thesis have been published previously in the following papers:

- [121] Liam Steadman, Nathan E Griffiths, Stephen A Jarvis, Stuart McRobbie, and Caroline Wallbank. 2D-STR: Reducing Spatio-temporal Traffic Datasets by Partitioning and Modelling. In *5th International Conference on Geographical Information Systems Theory, Applications and Management*, pages 41–52, 2019
- [122] Liam Steadman, Nathan Griffiths, Stephen Jarvis, Mark Bell, Shaun Helman, and Caroline Wallbank. Reducing and Linking Spatio-Temporal Datasets with kD-STR. In *3rd ACM SIGSPATIAL Workshop on Advances in Resilient and Intelligent Cities*. ACM, 2020
- [123] Liam Steadman, Nathan Griffiths, Stephen Jarvis, Mark Bell, Shaun Helman, and Caroline Wallbank. kD-STR: A Method for Spatio-Temporal Data Reduction and Modelling. *ACM Transactions on Data Science*, 2(3), 2021



# Abstract

Analysing and learning from spatio-temporal datasets is an important process in many domains, including transportation, healthcare and meteorology. However, in recent years, the volume of data generated for such datasets has increased significantly. This poses several challenges for data scientists, including increased processing overheads and costs. Thus, several methods have been proposed for reducing the volume of data stored and processed to analyse and learn from these datasets. However, existing methods fail to take advantage of the spatial and temporal autocorrelation present in spatio-temporal data, incur unnecessary overheads when retrieving the data, or fail to retain information about all instances and features.

This thesis introduces several data reduction methods to address these limitations. First, the  $k$ D-STR algorithm is introduced, which hierarchically partitions and models the data, thereby reducing the storage overhead of the dataset. This method minimises the storage used and error incurred. Second, this reduction method is adapted for the context of data linking, and an alternative heuristic proposed that minimises error in the features engineered during linking. Third, adapted algorithms are presented for reducing multiple datasets simultaneously, and reducing large datasets in a distributed manner.

Through empirical analysis using real-world datasets, the utility of these algorithms is investigated. The results presented demonstrate the data reduction that can be achieved using these algorithms, as well as the impact of using different spatial referencing systems and modelling techniques. Further analysis is presented that demonstrates the effect of error in location and time, noise and missing data on the data reduction. Combined, the algorithms presented offer an improvement over the state-of-the-art in spatio-temporal

data reduction, and the analysis presented demonstrates the results that may be achieved for datasets exhibiting a range of characteristics.

# Sponsorships and Grants

The research presented in this thesis was made possible by the support of the following benefactors and sources:

- Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training in Urban Science (EP/ L016400/ 1). 2016-2020.
- TRL Limited, Crowthorne House, Wokingham, UK, RG40 3GA. 2016-2020.

# Acronyms

**DCT** Discrete Cosine Transform.

**DTR** Decision Tree Regression.

**GPS** Global Positioning System.

**kNN** k Nearest Neighbours.

**LES** Linked Error and Storage.

**LRS** Linear Referencing System.

**MAPE** Mean Average Percentage Error.

**MAR** Missing at Random.

**MIDAS** Met Office Integrated Data Archive System.

**MNAR** Missing Not at Random.

**NRMSE** Normalised Root Mean Squared Error.

**PLR** Polynomial Linear Regression.

**RES** Reconstruction Error and Storage.

**SRS** Spatial Referencing System.

**WGS 84** World Geodetic System 1984.

# Notation

$\mathbb{S}$	Spatial Domain, which is $\mathcal{D}$ -dimensional
$\mathbb{T}$	Temporal Domain
$D$	A single spatio-temporal dataset
$S$	Set of sensor locations, with an individual location denoted $s$ or $s_i$
$T$	Set of timestamps at which sensors record instances, with an individual timestamp denoted $t$ or $t_i$
$T_{s_i}$	Set of timestamps at which the sensor at location $s_i$ recorded instances
$k$	Number of spatio-temporal dimensions
$d_{s,t}$	An instance in $D$ , recorded at location $s$ and time $t$
$\tilde{d}_{s,t}$	An imputed instance used in data linking scenarios
$\hat{d}_{s,t}$	A reconstructed instance at location $s$ and time $t$
$F$	Set of features of a dataset, with a single feature denoted $f$
$\delta_{\mathbb{S}}(s_i, s_j)$	Spatial distance between locations $s_i$ and $s_j$ , sometimes referred to as $\delta_{\mathbb{S}}(i, j)$
$\delta_{\mathbb{T}}(t_i, t_j)$	Temporal distances between times $t_i$ and $t_j$ , sometimes referred to as $\delta_{\mathbb{T}}(i, j)$
$\delta_{\mathbb{ST}}(u, v, w, x)$	Combined spatio-temporal distance between locations $u$ and $w$ , and times $v$ and $x$
$\delta_F(d_{s,t}, d_{u,v})$	Distance in feature space between instances $d_{s,t}$ and $d_{u,v}$
$P$	Set of partitions, with a single partition denoted $p_i$
$B$	Set of bounding polygons in the spatial domain, corresponding to the set of partitions $P$
$t_i^b$	Beginning timestamp of partition $p_i$
$t_i^e$	Ending timestamp of partition $p_i$
$D_{p_i}$	Subset of instances in $D$ that lie within partition $p_i$
$M$	Set of models, with a single model denoted $m_i$
$C$	Set of clusters, with a single cluster denoted $c_i$
$n$	Number of supplementary datasets used to augment a primary dataset
$D^P$	A primary dataset in a linking scenario

$D^l$	A supplementary dataset in a linking scenario
$D^{\text{Aug}}$	A primary dataset after it has been augmented using one or more supplementary datasets
$D_*^{\text{P}}$	A subset of the primary dataset $D^{\text{P}}$
$\hat{D}_*^{\text{Aug}}$	A subset of the primary dataset $D^{\text{P}}$ after it has been augmented using one or more reduced supplementary datasets
$F_{\text{aug}}^l$	The set of features engineered during the linking process for a primary dataset
$P^{\text{P}}$	Set of partitions belonging to the reduced primary dataset
$M^{\text{P}}$	Set of models belonging to the reduced primary dataset
$L$	Set of instances from supplementary dataset $D^l$ used to augment a primary dataset

# Chapter 1

## Introduction

For centuries, analysing phenomena through captured data has helped mankind progress. In many domains, a fundamental component of such analysis is understanding the spatial and temporal nature of the data. An early example demonstrating the importance of spatial location is John Snow’s 1854 analysis of a cholera outbreak in Soho, London [118]. After collecting data from households in the area and visualising the locations of cholera occurrences on a map, Snow noticed that many people with symptoms lived within the same neighbourhood. Snow’s breakthrough was understanding the significance of location in his data — at the centre of the neighbourhood was a water pump that Snow identified as the source of the outbreak. This breakthrough resulted in several analytical methods that started a new era of spatial and temporal data analysis, and earned Snow the informal title ‘*father of modern epidemiology*’.

Datasets such as Snow’s cholera dataset are referred to as *spatial datasets*. Similarly, datasets that record the time at which each measurement was taken are referred to as *temporal datasets*, and datasets containing both the spatial location and time for each measurement are referred to as *spatio-temporal datasets*. A unique quality of spatial, temporal and spatio-temporal data is the presence of dependencies between measurements taken close together in space and time. This is referred to as *spatio-temporal autocorrelation*, and its presence in data has aided progress in many domains including climate science, epidemiology, transportation, criminology and communications. Such datasets record specific variables, or *features*, about a phenomena or process. Furthermore, individual measurements of these features are referred to as *instances*, and the location and time at which each instance was recorded are also stored as features in the dataset. For example, in a dataset of vehicle counts for a road, each instance may refer to an individual vehicle and the features include the size and brand of the vehicle, as well as where and when the vehicle was seen. Alternatively, each instance may refer to a period of time defined by a start time, end time and location with a total vehicle count

feature. While some spatio-temporal datasets are created by hand, such as the dataset used by Snow, many are generated by sensors that record instances automatically.

In recent years, the quantity of spatio-temporal data generated and processed globally has grown significantly. The global value of the geospatial and spatio-temporal analytics industry is estimated to reach \$439.2 billion by the end of 2020, growing at an annual compound rate of 13.8% [55]. In part, this growth has been driven by the falling cost of sensors and data collection methods, and the more precise analysis they support. For example, in the transportation domain, more affordable sensors have enabled monitoring systems for transport networks to operate at a higher resolution in both space and time. However, the growth in spatio-temporal data has also been driven by a substantial growth in the global population and the expansion of urban environments. By 2030, it is expected that 5 billion people will live in urban spaces and, by 2050, over 68% of the global population will live in urban environments [130]. Such increases in population have created a greater need for the monitoring and prediction of resource usage in a range of sectors, again increasing the quantity of spatio-temporal data generated, stored and analysed.

Such increases in data volume pose two significant problems for data analysts. First, larger datasets require more storage. Though the cost per gigabyte (GB) of storage continues to decrease, the rate of data generation can outpace the rate of decrease in storage cost. Second, larger datasets require more computation time and memory to process, often rendering tasks infeasible that were once possible. In data linking tasks, in which one dataset is augmented with information from one or more other datasets, the amount of memory required can grow quadratically as the dataset sizes increase<sup>1</sup>. While a sensible approach for overcoming these issues may be to increase the number of processors used to process the data, this may be too expensive and these resources may not be available. Yet, the information provided by the data is still important and necessary — the higher spatio-temporal resolution of large datasets benefits areas of industry and research such as decision support and spatio-temporal data mining. There is therefore a need to reduce the storage overhead of spatio-temporal datasets and speed up their processing and analysis without compromising the information provided by the data.

A common method for reducing the storage overhead of spatio-temporal datasets is to compress the data. Many compression algorithms are well established and are able to compress datasets to representations that are significantly smaller than the original size. However, compression methods do not permit analysis to be performed on the compressed form of the data and require it to be decompressed before it can be used. An alternative method for

---

<sup>1</sup>In this thesis, the terms *linking datasets* and *augmenting datasets* are used interchangeably.



reducing the storage overhead of the data is to remove instances or features from the data, possibly replacing them with representative instances and features that aim to capture as much of the original information as possible. While this reduces the storage overhead of the dataset, it can remove information that may be important in later analysis. Furthermore, replacing the original features with engineered features can require further transformations back to the original feature space before the data can be analysed.

Other methods for reducing the storage and processing overheads of spatio-temporal datasets include methods that reduce the data to a series of models, a process known as dataset *reduction*. These methods split the dataset into sets of instances and features and store one or more models for each split. In common with feature engineering techniques, reduction techniques are able to support approximate data mining in which an approximate answer to a question is acceptable if the answer can be computed using less time, energy or memory than it would using the raw data. However, existing data reduction methods fail to take advantage of the large areas and time periods of high autocorrelation when partitioning the data. Furthermore, these methods focus on reducing the storage overhead of the dataset, increasing the processing required to extract raw data from the reduced form and, in some cases, preventing analysis on the reduced form.

Thus, a new data reduction method is required that overcomes these issues. First, such a method should use the variability of the data in space and time to partition the data into sets of similar instances. Second, the method should retain information about all instances and features while reducing the quantity of storage used by the reduced form of the data. Third, the method must allow data to be retrieved efficiently, without requiring multiple steps of processing or decompression to reconstruct or impute instances. Finally, the method should allow some analysis to be performed on the reduced form of the data, removing the need for many instances to be reconstructed where possible.

This thesis therefore addresses the need for a novel spatio-temporal data reduction technique that minimises the storage overhead of the reduced data and the computational overhead of retrieving the raw data from the reduced form. Specifically, this thesis lies within the domain of methods for spatio-temporal data science, alongside the domains of spatial databases and data compression, and builds upon work from the clustering and modelling domains. The aim is to use model-based reduction to reduce the storage overhead of spatio-temporal data while permitting analysis on the reduced form.

A novel method for reducing spatio-temporal data is presented, as well as several adaptations that offer improvements for common scenarios. The offline reduction of spatio-temporal data is considered, wherein the complete raw data is available as input. This work is motivated by data collected in the

transportation domain, as well as data that is commonly used to contextualise transportation data. The data used throughout this thesis is real-world data that exhibits varying degrees of spatio-temporal correlation, noise and missing instances, and illustrates the results that might be achieved with similar datasets.

## 1.1 Objectives

In the context of this thesis, the aim of reducing spatio-temporal datasets is to reduce the storage overhead of a dataset while permitting some analysis on the reduced data. Furthermore, the error incurred when reconstructing the instances from the reduced form must be minimised, and information about all features and instances retained. More precisely, the objectives of spatio-temporal reduction are as follows:

1. **To minimise storage used by the reduced dataset**

In many applications, the quantity of data present makes the processing of spatio-temporal datasets impractical. The aim is therefore to minimise the quantity used to store the reduced dataset.

2. **To minimise loss of information in space and time in the reduced dataset**

As well as reducing the storage overhead of a dataset, minimising the loss of information is equally important. That is, information should be stored about the entirety of the spatial and temporal domains covered by the original dataset, as well as the features present.

3. **To maximise accuracy of representation**

In data retrieval tasks, accurate imputation of the feature values in space and time is important. That is, the error between the instances in the original dataset and the instances reconstructed at the same locations and times from the reduced dataset should be minimised. Furthermore, it is important to accurately impute instances using the reduced dataset that were missing from the original dataset. Thus, the difference between the original instances and the reconstructed feature values at the same locations and times should be minimised, and the reduced dataset should also facilitate imputing missing data as accurately as possible.

4. **To permit analysis using reduced representation**

Methods such as compression algorithms can reduce the storage overhead of a dataset while incurring minimal or zero error. However, these methods do not allow analysis to be performed on the reduced dataset. That is, the original data must be decompressed before analysis can

be performed, and some methods require many or all of the original instances to be decompressed to retrieve a single instance. This increases the memory required to analyse the dataset as a single instance cannot typically be retrieved with random access. Instead, the aim of spatio-temporal reduction is to permit some analysis on the reduced form and permit the imputation of a single instance without requiring other instances be imputed first.

Together, these objectives define the aims of this research: to reduce a spatio-temporal dataset to a form that has a smaller storage overhead and requires less processing, while minimising the error incurred and the information lost.

## 1.2 Contributions

In Chapter 2, existing methods and the state-of-the-art that partially fulfil the objectives discussed above are presented. However, as we discuss in Section 2.6, the state-of-the-art methods fail to meet all of these objectives. Therefore, this thesis presents a novel reduction method for spatio-temporal datasets that meets the objectives set out above. Furthermore, this thesis presents adaptations of this method for data linking scenarios, reducing multiple datasets simultaneously and reducing them in a distributed manner, as well as an analysis of the robustness of the method to noise and missing data. Specifically, the contributions of this thesis are as follows:

- In Chapter 4, a novel reduction algorithm for spatio-temporal datasets,  $k$ D-STR, is presented. A novel partitioning technique is developed that uses the similarity of instances in space and time to partition a dataset. This partitioning technique overcomes shortcomings in state-of-the-art techniques. By fitting a model to the instances within each partition, and storing the coefficients of each model,  $k$ D-STR is able to store the dataset in this reduced format. In Chapter 4,  $k$ D-STR is evaluated using a range of datasets exhibiting different spatio-temporal characteristics.
- In Chapter 5, an alternative heuristic function for  $k$ D-STR is presented for data augmentation scenarios. When reducing datasets, this alternative heuristic considers the error introduced during dataset linking rather than during the reconstruction of the original features. Through experimentation, the utility of this alternative heuristic is demonstrated, as well as the speed-up achieved by reducing the datasets before linking.
- In scenarios where multiple datasets with similar coverage of the spatial and temporal domains are to be reduced, it can be more efficient to reduce

them simultaneously rather than individually. Therefore, in Chapter 6, an adaptation of the  $k$ D-STR partitioning scheme is presented for reducing multiple datasets simultaneously. This overcomes a weakness of existing methods for partitioning multiple datasets. Moreover, the impact of reducing datasets simultaneously on the reduction output is examined.

- Furthermore, in scenarios where a dataset’s volume is too large to be reduced on a single machine, distributed computing environments are used. However, it is ambiguous how  $k$ D-STR should partition and model datasets in such environments. Therefore, Chapter 6 also presents an adaptation of  $k$ D-STR that overcomes these issues.
- Finally, the robustness of  $k$ D-STR to erroneous and missing data is also examined in Chapter 6. An empirical assessment of  $k$ D-STR’s robustness to spatial and temporal error is presented, as well as  $k$ D-STR’s robustness to noise in the feature values and missing data. These experiments demonstrate how the output of  $k$ D-STR is affected as the amount of error or perturbation increases, and suggest what results can be expected for datasets with similar characteristics to the perturbed datasets tested.

This thesis is structured as follows. Chapter 2 provides a full description of the background research relevant to this thesis. Chapter 3 presents the metrics used to describe spatio-temporal datasets and introduces the datasets used throughout this thesis. Chapter 4 presents a novel spatio-temporal reduction method, namely the  $k$ -Dimensional Spatio-Temporal Reduction ( $k$ D-STR) algorithm. Chapter 5 describes the common task of linking spatio-temporal datasets and presents a methodology that links and reduces datasets. Chapter 6 presents an adaptation of the  $k$ D-STR data reduction method for reducing multiple datasets simultaneously, as well as an adaptation for reducing datasets in a distributed manner. Furthermore, Chapter 6 examines the effect of noise and missing data on the reduced data. Finally, Chapter 7 discusses the contributions of this work, and Chapter 8 concludes this thesis and suggests potential future research opportunities.

## Chapter 2

# Background

Spatio-temporal datasets are used in a wide range of fields to model and analyse complex and evolving processes. However, as the volume of spatio-temporal datasets continues to increase, analysing these datasets requires more computation and memory, and the task can sometimes become infeasible entirely. Therefore, data reduction techniques are used to reduce the quantity of data stored and processed for analysis tasks. However, existing techniques have several shortcomings that may result in less efficient reductions or remove information about some instances and features.

In this chapter, common tasks for analysing spatio-temporal data are presented, as well as existing techniques for reducing spatio-temporal datasets. First, Section 2.1 introduces motivating examples of common data analysis tasks. In Section 2.2, existing techniques for reducing datasets are presented, including techniques that partition and model the data. In Section 2.3, existing methods for partitioning spatio-temporal datasets are presented, and in Section 2.4 common methods for efficiently modelling spatio-temporal data are discussed. Section 2.5 presents techniques found in literature for analysing spatio-temporal algorithms. Finally, Section 2.6 discusses the shortcomings of existing data reduction techniques and the motivation for this thesis.

## 2.1 Spatio-Temporal Analysis Tasks

Methods for learning from and analysing spatio-temporal data are used in a wide range of industries and domains. In this section, an overview is presented of common analysis tasks that motivate this thesis.

### 2.1.1 Analysing and Learning from Spatio-Temporal Data

A common task when learning from spatio-temporal data is to group instances based on their similarity or proximity in space and time — a task known as cluster analysis. When instances are clustered in the spatio-temporal domain,

hot-spots of events are found, such as accident hot-spots in a road collision dataset. Many methods for clustering are developed from spatial statistics methods such as the spatial scan statistic [74], and generalisations for spatio-temporal data have been explored for topics such as disease outbreaks [44, 149]. When instances are clustered in the combined spatio-temporal domain and feature space simultaneously, hot-spots of *similar* events are detected. Many of these methods are derived from the DBSCAN algorithm [48], such as ST-DBSCAN [20]. This algorithm uses two distance metrics (one in the spatial domain and one in the combined temporal domain and feature space), and uses separate thresholds for the two metrics to define whether any pair of instances should be placed into the same cluster.

As well as clustering individual instances, clustering methods for trajectories have been developed, such as Trasarti *et al.*'s two-step clustering method for clustering movement patterns of people [128]. Clustering methods have also been proposed for spatial time series data and spatial maps recorded at different times. These have a similar aim to pattern mining, which aims to find similar subsets of instances that repeat in the spatial and/or temporal domains. In particular, motif discovery aims to find repeating patterns of sequential instances in time series recorded at different sensors. Such methods have been used and studied extensively in applications including ecology, medicine and finance [91, 127].

A second common task is predicting future instances in datasets collected from sensors, although many approaches in literature consider each sensor as an independent time series and ignore the spatial dependency between sensors. Variants of neural network approaches that consider the spatial dependencies between sensor time series' have been explored in literature [67, 132], although there is a need for further research in this area [13]. Similar to prediction, instance imputation estimates the feature values of the data at times and locations not present in a dataset. For example, approaches for imputing data have been studied heavily in remote sensing [99] and epidemiology [100]. Many of these methods use spatio-temporal adaptations of spatial prediction methods from statistics literature [38], although other techniques use nearest neighbour and neighbourhood methods (described further in Section 2.1.2). A further common task is detecting anomalous data or *outliers*. A number of techniques have been proposed for detecting anomalous instances in the temporal domain [59] and spatial domains [2], although further research is needed for detecting spatio-temporal outliers [13, 139]. One proposed approach is to use ST-DBSCAN [20] to detect outliers, although this assumes homogeneity in neighbourhood properties across space and time [13].

Each of the algorithms used for these analysis tasks are aided by the spatio-temporal correlation present in the data. In particular, the clustering of similar

instances relies on finding regions of similar instances in the data, and pattern discovery aims to find recurring groups of instances in the data. Prediction and imputation methods rely on the correlation between nearby instances to infer what the feature values at an unsampled point would be, and the same correlations are exploited by anomaly detection algorithms for classifying instances as unexpected or erroneous.

### 2.1.2 Linking Spatio-Temporal Datasets

Often, data scientists wish to analyse multiple datasets in the context of each other. For example, instances in one dataset may be linked to instances in other datasets by their proximity in space and time. In another scenario, instances in one dataset may be augmented by the instances in one or more other datasets. This is referred to as relationship mining or *dataset linking*, and is a common task in spatio-temporal data science.

#### Co-Occurrence Mining

Events recorded in a spatio-temporal dataset may be semantically related if they occurred at nearby locations and times, and identifying such events is referred to as co-occurrence mining. A simple approach for detecting co-occurring events is to segment the spatial and temporal domains using regular grid partitioning, and link any events or instances that reside within the same partition. However, this approach is sensitive to the chosen grid size, and so a more common approach is to use points of interest to inform the partitioning of the spatial domain. For example, Zhang *et al* used this approach to link police complaints in New York City to taxi ride, bike rental, street information and social media data [145]. To segment the spatial domain, major roads within the city were used as boundary lines of regions, thus maintaining the semantic meaning of neighbourhoods within the partitioning. To partition the temporal domain, instances were binned into 30 minute segments.

Other examples of partitioning and binning methods for co-occurrence mining include Kong *et al.*'s analysis of taxi trips, points of interest and events data in Shanghai [73]. In their analysis of traffic data, Ding *et al.* used Voronoi partitioning to partition the spatial domain [46]. The intersections of roads were used as the seeds of the Voronoi partitioning, although only intersections that were more than 50 metres apart were considered to prevent each Voronoi partition being too small to contain a traffic sensor. Instances in one of the datasets being linked were defined by a spatial area rather than single point location in space, and so each instance in the dataset was linked to a partition if their spatial areas overlapped. Such datasets are referred to as *field datasets* (discussed further in Section 3.1), and approaches for linking such datasets

include the Dimensionally Extended nine-Intersection Model (DE-9IM) [133] for the spatial domain, and Allen’s model for the temporal domain [8]. These models formalise spatial and temporal predicates for field data, such as *equals*, *contains*, *intersects* and *overlaps*.

### Neighbourhood Linking Techniques

While partitioning methods for co-occurrence mining allow for efficient instance linking, they fail to link instances that are close in space and time but are placed into different partitions. To overcome this, neighbourhood methods link instances in multiple datasets that are within a fixed spatial and temporal distance from each other. This approach is commonly used when augmenting the instances in one dataset with instances from one or more other datasets. For example, the Australian Urban Research Infrastructure Network (AURIN) uses neighbourhood methods for augmenting datasets for tasks such as investigating the link between consumer spending on alcohol and a person’s access to alcohol [115]. In this study, the algorithm used neighbourhoods to calculate the number of survey respondents that lived within a fixed distance of licensed alcohol premises in Victoria, Australia. Since radial neighbourhoods, which the authors note are commonly used in augmenting tasks, do not reflect the true walking distance between each respondent and premises, the local road topology was used when calculating the neighbourhood perimeter.

In scenarios such as the AURIN investigation, all instances within the neighbourhood are considered to be equally important to the instance at the centre of the neighbourhood. However, in some scenarios the distance of an instance from the centre of the neighbourhood is important. For example, Knittel *et al.* used inverse distance weighting (IDW) to weight nearer instances more highly when augmenting a mortality dataset with traffic, weather and pollution data [71]. The IDW approach was used to more accurately estimate the pollution and traffic incident on each mortality instance. Spatial and temporal neighbourhoods have also been used in other spatio-temporal analyses, including studies on pollution [5].

### Nearest Neighbour Linking Techniques

When the distribution of instances in the spatial and temporal domains of a dataset is highly unbalanced, some instances may be linked to many other instances in a fixed-size neighbourhood, while others may be linked to few. In such cases, choosing the nearest  $k$  neighbours to an instance may be more beneficial — a method known as  $k$  Nearest Neighbour linking (kNN). For example, Shen and Masada used the kNN method to calculate the distance from each rental apartment in a New York City dataset to its nearest 5 instances



of different types of amenity [112]. For each apartment, the minimum, mean and maximum distance to each amenity was calculated and added as features to the instance.

In the road accident analysis domain, Theofilatos *et al.* augmented road accident instances with weather and traffic information by linking each accident to its nearest instance in a weather dataset and traffic dataset (i.e.  $k = 1$ ) [126]. Domain-specific restrictions were used to select the most appropriate instances for linking, for example using the nearest traffic sensor prior to the accident’s location on the road. Furthermore, neighbourhood restrictions were also used to limit the maximum temporal distance between each accident and its nearest weather and traffic instances. If no instance was found within that limit, the accident was removed from their analysis. In other work, Sathiaraj *et al.* found IDW nearest neighbour linking to be more accurate when imputing precipitation, temperature and visibility features compared to ordinary kriging and radial basis function (RBF) methods [109].

## 2.2 Reducing Spatio-Temporal Datasets

The quantity of data present in many spatio-temporal datasets makes them difficult or infeasible to process in their raw forms. This presents issues for data science and analysis tasks, such as those presented in Section 2.1. To facilitate faster processing, many techniques exist that reduce the quantity of data that needs to be processed. These techniques aim to minimise the difference between the analysis and models created using the reduced dataset and the raw dataset. In this section, existing methods for reducing the quantity of data to be processed in a spatio-temporal dataset are presented.

### 2.2.1 Feature Selection Techniques

Many existing methods for reducing datasets focus on removing a subset of features, thereby reducing the quantity of data used to store each instance. These are referred to as feature selection techniques, and can be split into three categories. The first category, filter methods, rank features according to a relevance criterion, such as Shannon entropy, and remove any features that have a relevance below a predefined threshold. However, since filter methods are independent from the learning and analysis tasks the data is used for, they can remove features that are may be relevant in those later tasks. Furthermore, choosing a threshold of relevance for the features is dataset specific and can be time consuming.

The second category of techniques, wrapper methods, use search algorithms to find the optimal subset of features according to an objective function. For

example, a classifier may be used with labelled data and the subset of features chosen that maximises the accuracy of predicting a class label in the data, although training that classifier may not be the users' intended task for the data [111]. The third category of feature selection techniques, embedded techniques, perform feature selection in the process of model training. In contrast to wrapper methods, the model trained by an embedded technique is the model output and used by an analyst.

Several feature selection techniques for real-valued datasets have been evaluated in the context of different domains. A comparison of the FOCUS [9] and RELIEF [70] filtering methods found that both methods yield similar accuracy rates for Support Vector Machine, Naïve Bayes and kNN classification compared to using the full dataset [89]. Yet, the time required to process the data is notably lower. In a similar evaluation, correlation-based feature selection was found to yield highly accurate classifiers using just  $\approx 6\%$  of the original features [33]. Several other feature selection techniques for real-valued data exist, and a number of reviews of these can be found in the literature [82, 111, 119].

While feature selection techniques reduce the quantity of data stored per instance, they may remove features that are significant for future analysis and may fail to capture the information present in the data [66]. Furthermore, they typically fail to take advantage of spatio-temporal correlation, meaning that a more efficient reduction may be possible.

### 2.2.2 Feature Extraction Techniques

In contrast to feature selection, feature extraction/engineering methods project the original features of a dataset onto a new feature space. Often, the new feature space has fewer dimensions, thereby reducing the number of values stored for each instance. The best mapping between the feature spaces is that which optimises an objective criterion, such as explained variance or accuracy, when combined with modelling. Linear feature extraction algorithms include Principal Components Analysis (PCA) [98] and Linear Discriminant Analysis (LDA) [88]. While PCA maximises the variance between features in the new feature space, LDA minimises the variance within a class and maximises the variance between classes. For spatio-temporal data, assumptions made by PCA often require adaptations to account for the correlations between near instances in space and time, and a discussion of these adaptations can be found in literature [42].

Non-linear feature extraction algorithms, sometimes referred to as manifold learning, map high-dimensional datasets to lower dimensions such that the mapping reflects the structural features of the data. For example, the Isomap

method [15] uses a geodesic distance measure between instances, and the Locally Linear Embedding (LLE) method [106] improves on Isomap by reducing the computation required. A review and comparison of these techniques can be found in literature [84].

Feature extraction techniques offer an improvement over feature selection techniques by minimising the amount of information or variance lost when reducing a dataset. However, they may still fail to capture all of the variance in the original dataset, and it may be more beneficial to retain information about all of the features. Furthermore, they require a mapping back to the original feature space for many types of analysis, thereby removing the speedup achieved by reducing the number of features. Instead, methods that take advantage of the correlations and patterns present while retaining all features in the original dataset may be more efficient.

### 2.2.3 Instance Selection and Abstraction Techniques

Instance selection techniques retain a subset of the original instances in a dataset that are sufficient for a given task. For example, for classification tasks, only those instances that are required for accurately classifying unseen instances are retained. Several established algorithms exist for instance selection, such as the IB3 incremental algorithm [3]. In IB3, a set of selected instances  $\mathcal{S}$  is initialised to contain a single instance chosen from the set of input instances at random. Then, each instance in the dataset is considered in turn. If an instance  $x$  is correctly classified by its nearest neighbour in the feature space from  $\mathcal{S}$ , instance  $x$  is disregarded and not added to  $\mathcal{S}$ . However, if  $x$  and its nearest instance in  $\mathcal{S}$  have different class labels,  $x$  is added to  $\mathcal{S}$ . Thus, after all instances have been considered,  $\mathcal{S}$  contains the instances that are sufficient for classifying instances similar to those in the input dataset. In the spatio-temporal domain, Whelan *et al.* have used k-medoids clustering to reduce a dataset to  $k$  instances [136].

Instance abstraction techniques, like feature engineering techniques, create a smaller set of *prototype* instances that represent the original instances but are not necessarily present in the input dataset. Abstraction techniques have been shown to reduce the number of instances required for tasks such as k-nearest neighbour classification, and training models on these reduced datasets is demonstrably faster with minimal effects on classification accuracy [96]. One example, the Prototypes for Nearest Neighbour (PNN) algorithm, is a supervised method that iteratively creates weighted prototypes that achieve approximately the same classification accuracy as the original dataset [27]. Another example, the Decision Surface Mapping (DSM) algorithm, selects instances to be prototypes at random from the original dataset, and moves those

selected instances in the feature space to improve the classification accuracy on the rest of the dataset [56]. The Learning Vector Quantisation (LVQ) family of algorithms operate in a similar fashion to the k-means algorithm [72]. Rather than updating prototypes only when instances are misclassified, the LVQ algorithm also updates prototypes when instances are correctly classified. Comparisons of instance abstraction techniques can be found in literature [129], and examples of the use of instance abstraction techniques for spatio-temporal data can be found in literature [135].

Similar to feature selection and extraction techniques, instance selection and abstraction techniques remove instances that may be significant in later processing and analysis. They also reduce the variance of the dataset and may not accurately capture outliers that are of interest in later processing, and querying individual instances may no longer be possible. Thus, techniques such as these restrict the analysis that can be performed on the reduced data they output.

#### 2.2.4 Data Sketching Techniques

Selection and engineering/abstraction techniques focus on removing or prototyping instances and features. In contrast, sketching techniques create summaries of the data that are query-specific using a limited number of passes over the data. In doing so, a reduced form of the data is created that can be stored and processed more efficiently than the original data which, in the context of data streams, may no longer be available. Many sketching techniques focus on counting items, such as the Count-Min sketch and its adaptation for real-valued data [36, 116]. Another, the Bloom filter and its variants answer questions about the membership of an item to a set using hash tables [22]. The Hyper-LogLog (HLL) algorithm uses a probabilistic counter to answer cardinality questions and is sufficiently efficient to be used with very large quantities of data [50]. Furthermore, methods such as Min-Hash find the approximate similarity between two items using their Jaccard similarity and hashes of the items [25]. Reviews of sketching methods can be found in literature [113]. However, these techniques do not consider the spatial and temporal nature of the data and require the user to have knowledge of the type of analysis they will use the reduced data for in advance.

In the spatio-temporal domain, a method has been proposed that combines instance selection sketching with the Kalman filter to track large-scale spatio-temporal processes [18]. In another example, Tai *et al.* presented a sketching method for building linear classifiers over a spatio-temporal dataset [125]. By building linear classifiers over the temporal streams of a set of sensors, correlated features are identified while permitting analysis of the stream's

instances. However, this methodology destroys features which are not heavily weighted by the linear classifier.

Overall, sketching techniques are highly specialised to the type of analysis they permit, but this prevents their output from being used to answer other questions and analyses [37, 113]. They are created for specific queries and, since the original dataset is destroyed after the sketch is created, it is not possible to reconstruct the raw data for other purposes. Most techniques do not take advantage of the spatial and temporal correlations in the data, and many require knowledge of which features or instances will be of interest before the sketch is created, which may not be known ahead of time.

### 2.2.5 Reducing using Modelling Techniques

While the techniques discussed thus far result in the loss of instances/features, or make the original data unrecoverable, some techniques exist for reducing datasets using modelling techniques. These can be split into techniques that partition datasets into subsets or *blocks* of instances and store a model per block, and techniques that store a model for the entire dataset. The IDEALEM algorithm for temporal data streams partitions the instances of a dataset into fixed size blocks, and uses key statistical properties calculated over these blocks to identify blocks that are statistically similar [78, 79]. Statistics used by IDEALEM include the minimum, maximum and average values for each feature. For each set of statistically similar blocks, the raw data of one block is retained along with summary statistics and where the block repeats in the time series. By analysing each of its prototype blocks, IDEALEM allows users to identify unusual temporal periods that do not fit expected trends. It also enables comparison of different time periods, retains information about all features, and allows for faster generation of statistics compared to the original dataset. However, by replacing instances with links to prototype blocks, IDEALEM removes entire subsets of instances. Furthermore, since the method does not consider the spatial nature of spatio-temporal data, IDEALEM does not permit spatial imputation without further modelling.

Similar to IDEALEM, the ISABELA algorithm partitions each feature of a dataset into fixed-size spatial windows [75]. The observed feature values within each window are then sorted into ascending order and a B-spline curve fitted to the data. By storing the parameters of the fitted curve with temporal encoding, the dataset can be reduced to a set of model parameters. However, ISABELA requires this reordering of instances as it is designed for data that is highly varied in space and time. This requires a mapping back to the temporal domain when retrieving the data, and is unnecessary for many types of real-world datasets that are more cyclical (such as traffic data). Furthermore,

this re-ordering makes imputation of instances for a given location and time impossible without first retrieving the full dataset and reordering back into temporal order. ISABELA does permit statistics to be calculated over the data if the temporal period of interest is exactly covered by one or more windows, otherwise mapping back into the temporal domain is again required. In the same way, identifying unusual spatial or temporal regions is partially supported.

A method that overcomes the limitations of IDEALEM has been proposed by Yang and Chen [142]. This method partitions a single time series into blocks and fits a non-linear regression model to each block, using the discrete time interval as the predictor variable and the feature value as the response variable. By storing the coefficients of the models, this method reduces the storage overhead of the data and permits imputation from the models. However, the technique requires the user to select the number of blocks and coefficients stored from the model, and these parameters greatly affect the storage used and error incurred. Furthermore, the method does not consider the spatial domain of the data. A similar method that uses a linear model for each partition has been proposed for scientific data [4].

In the category of techniques that store a model for the entire dataset, deep autoencoders have been used to model the temporal features of spatio-temporal datasets [131]. The Sparse Autoencoder (SAE) has been used to reliably estimate missing data in spatio-temporal datasets [137]. This fitting of a summary, which minimises the root mean squared error (RMSE) over instances in the discrete spatial and temporal domains, is able to impute missing values given other instances from the same time interval. This approach may be adapted to incorporate multiple time intervals, e.g. the whole dataset, and store the autoencoder weights for the purposes of storing and reconstructing the dataset. However, autoencoder weights are incomprehensible in analysis and so prevent manual analysis of the reduced dataset.

In the domain of traffic dataset analysis, Pan *et al.* proposed a two-stage algorithm that summarises spatio-temporal traffic sensor datasets [97]. This method creates a signature of the dataset in the spatial and temporal domains using a technique such as wavelet decomposition, and a set of outliers that fall outside an acceptable error margin of this signature. While this technique is good at capturing the cyclic and seasonal natures of many datasets, it performs poorly on datasets containing irregular patterns or many outliers. For example, in road traffic data, instances from national holidays (temporal domain) and areas of construction work (spatial domain) are known to deviate from regular traffic cycles, and so are labelled as outliers. Pan *et al.*'s algorithm will only retain some of these outliers due to its probabilistic nature, and so reconstruction of these instances may be highly inaccurate.

## 2.3 Partitioning Datasets

Methods such as those presented in Section 2.2.5 partition and model the instances within a dataset to reduce the quantity of data stored. Partitioning methods for spatio-temporal data are used in many fields of computer science and data science. For example, in distributed computing, the Columbus framework hashes the timestamp of each instance to distribute instances between compute nodes, then uses a grid partitioning of the spatial domain on each processor to store the instances [11]. This partitioning scheme speeds up answering queries for a given location and time by reducing the number of instances that need to be processed. In spatio-temporal reduction, a similar method has been used in multiple works including IDEALEM [78, 79], ISABELA [75] and the technique proposed by Yang and Chen [142]. However, these methods are sensitive to the size of grid cell or block used: using many small partitions requires many models to be stored, and many of these models may be similar and therefore redundant; using many large partitions increases the variance within each partition and requires many coefficients to be stored per model. In distributed computing, methods have been proposed that overcome this issue by automatically splitting a partition into smaller partitions when the number of instances it contains reaches a threshold. For example, the GeoBeam framework [61] makes use of Quad-Trees [54] for 2-dimensional domains and KD-Trees [17] for partitioning the data in  $k$  dimensions. When a partition is to be split, these methods bisect the partition along the median location in each dimension, thereby ensuring each new partition has approximately the same number of instances.

Yet, these methods partition the spatial and temporal domains according to the maximum number of instances that can be processed on a single compute node or according to a threshold defined by the user. Instead, more appropriate measures may be used to determine how to partition the data. For example, the MR-DBSCAN algorithm uses a heuristic to estimate the computation time required for computing the DBSCAN algorithm over the instances within a partition [60]. A partitioning of the data is then formed that minimises the maximum heuristic value of all of the partitions, thereby minimising the overall computation time. Similarly, the DBSCAN-MR algorithm partition uses a heuristic to minimise the communication overhead between compute nodes when computing the DBSCAN algorithm in a distributed computing environment [60]. Similar to median-based partitioning methods, these methods are not designed for data reduction and do not consider the amount of data required to store models of the data. That is, they do not consider the dissimilarity of the data in the feature space when partitioning the data in the spatio-temporal domain. As a result, some partitions may require many model coefficients to

capture the variability of the data accurately, and a more efficient reduction may be achieved by considering the feature values of the data.

One method that does support partitioning the data in the spatio-temporal domain while considering the dissimilarity of near instances in the feature space is ST-DBSCAN [20]. This density-based approach is based on DBSCAN, which uses a maximum distance threshold  $\epsilon_f$  to define separate clusters in the feature space [48]. ST-DBSCAN extends this approach with a maximum threshold  $\epsilon_s$  for the spatial domain and  $\epsilon_t$  for the temporal domain, thereby defining the maximum distance two neighbouring instances can be in both the spatio-temporal domain and the feature space in order to be connected. This algorithm allows the dissimilarity of instances to be considered when partitioning the data, however it does not consider the total variance within each partition. That is, the instances within a partition may be less than  $\epsilon_f$ ,  $\epsilon_s$  and  $\epsilon_t$  apart, yet the overall variance between all instances within the partition may be high, thereby requiring many model coefficients to accurately capture the data. Furthermore, ST-DBSCAN features the parameters  $\epsilon_f$ ,  $\epsilon_s$ ,  $\epsilon_t$  and minPts (the minimum number of instances within a partition) that must be fine-tuned by the user, thereby incurring the same issues as Quad-Tree and KD-Tree based approaches.

Thus, for methods that partition and model datasets for the purposes of reduction, existing partitioning techniques are limited in their usability. Instead, a partitioning method that considers the variance within each partition and does not use a regular partitioning in space and time is needed.

## 2.4 Modelling Spatio-Temporal Data

When modelling instances for reduction, the aim is to accurately capture the variance in the data over space and time while using significantly less storage than the raw data. Furthermore, the models formed should allow the original data to be retrieved using just the spatial and temporal locations of the instances as input. These requirements make many common modelling techniques for spatio-temporal data unsuitable for reduction. For example, deep learning methods have been proposed for imputing and predicting spatio-temporal data [52, 132]. In particular, several methods have been proposed recently for traffic estimation, wind speed and air pollution estimation [132]. However, deep learning methods require many coefficients (referred to as *parameters*) be retained in order to store a model. In some scenarios the number of coefficients stored may be higher than the storage volume of the original dataset, negating the benefits of reducing a dataset. Deep learning methods also require significant computation time to train models, further reducing their desirability for data reduction.



A second family of modelling techniques that are not suitable for reduction are spatio-temporal statistical modelling techniques [90]. Techniques in this family are used for imputing values in the data at times and locations not sampled, and include autoregressive moving average and kriging techniques. For example, Spatio-Temporal Kriging (ST Kriging) uses the semivariogram of the data, which measures how the data varies over space and time, to impute values in the data at locations and times not sampled [38]. For a given location and time that are not present in the original dataset, the feature values of the data are estimated using a weighted sum of the instances present in the data. However, these techniques require some of the data to be stored, potentially requiring significant quantities of the original data be stored to capture the variance of the data and support accurate imputation.

The most common family of modelling techniques for reduction in literature is regression. In some cases, linear regression has been used for compressing spatio-temporal data, however many spatio-temporal datasets do not follow linear trends. In their compression of earthquake data, Yang and Chen demonstrated that non-linear regression models achieve lower storage overheads compared to linear regression [142]. The proposed method specifically tested exponential and logarithmic models, with the spatial and temporal locations of each instance used as predictors of the feature values. In epidemiology, generalised linear mixed models are commonly used to model the spread of diseases such as Dengue Fever [12], although features such as *rainfall*, *humidity* and *population* are used as predictors alongside spatial and temporal location. For example, in [80], the population at a location  $i$  and its rainfall and temperature at time  $j$  are used as predictors. In other examples presented in [12], the spatial and temporal locations of instances are again used to impute the feature values of instances not present in the original data. Such models have the benefit of requiring few coefficients to be stored while capturing the majority of the variance in the data. Furthermore, they allow for imputation at locations and times not present in the data, and these properties make them ideal modelling techniques for reducing spatio-temporal data.

An alternative modelling technique to regression is cosine and wavelet transforms. In the wireless sensor networks domain, discrete cosine transforms (DCT) have been used to effectively compress time series data for communication between sensor nodes [28]. The DCT method expresses a time series as the weighted sum of cosine functions with different frequencies. By retaining only those weights that have a high absolute value, the number of coefficients stored or transmitted for the data is minimised while allowing the majority of the information in the data to be retained (i.e. the error incurred is minimised). DCT also allows for direct imputation of instances by inputting the desired location into the cosine functions of differing frequencies, and calculating the

weighted sum of their outputs to impute the feature values at that location. DCTs have also been shown to be effective at reducing other time series datasets, including electro-encephalographic (EEG) data [21], and the method may be adapted for spatio-temporal data by using multi-dimensional DCTs [103].

Both regression and cosine/wavelet modelling allow the user to vary the number of coefficients stored for a model. This property makes them particularly desirable for reduction as the accuracy of the model can be loosely controlled by varying the number of coefficients stored. When fewer coefficients are stored, a less accurate model is formed although this uses less storage, while a more accurate model is formed when more coefficients are stored, yet this comes with increased storage overhead.

## 2.5 Evaluating Spatio-Temporal Methods

Methods used for analysing and reducing spatio-temporal data are aided by the spatio-temporal correlation present in the data. For example, reduction techniques that model the data rely on nearby instances being similar to each other, thereby allowing them to be accurately captured by the same model. However, errors in the data can decrease the similarity of nearby instances. When the recorded locations of instances are inaccurate, or the feature values recorded are noisy, the spatio-temporal variance of the data may increase. In the context of data reduction, increased spatio-temporal variance requires more model coefficients to accurately capture the data, resulting in a less efficient reduction. Furthermore, missing instances may increase the inaccuracy values imputed from models. Thus, it is important to understand the robustness of spatio-temporal analysis and reduction methods to erroneous and missing data. The impact on a method of erroneous spatial and temporal locations, noise in the values recorded and missing data inform how a method should be used and what data it may not be appropriate for. Three types of empirical analysis are typically used to study the robustness of spatio-temporal methods: robustness to error in the reported location or time of an instance, robustness to noise in a datasets' features, and robustness to missing data.

### 2.5.1 Errors in Location of Sensors

The sensitivity of spatio-temporal methods to error in location and time can be tested through perturbations of the data. In the spatial domain, multiple studies have investigated the impact of spatial error on methods for visualisation [40], cluster analysis and event detection [53, 87], and aggregate statistical measures [58]. In all of these studies, *clean* or unperturbed data is used as a baseline and the methods being analysed computed on the clean data. Then,

each of the instances are moved by a distance drawn from a distribution to create an *unclean* or perturbed dataset, and the method recomputed for this data. The difference between the outputs of the method is then measured and analysed.

For example, in a study on the effect of geocoded location error, Malizia perturbed the locations of instances in a burglary dataset by distances of up to 200 m in a direction chosen from a uniform distribution [87]. This study tested the impact of location error on the space-time permutation scan statistic (STPSS), a measure for detecting hotspots or clusters of events in spatio-temporal data. The distances used were chosen from an exponential distribution as this best matched their empirical distribution of geocoding error. In another study, Griffith *et al.* perturbed the locations of soil samples in a uniformly random direction by fixed distances (e.g. 10 m) to test the impact on aggregate statistical measures [58].

Each of these studies used perturbations in space to test the robustness of cluster analysis, imputation methods and spatial statistics to errors in location. The distributions from which the perturbations were chosen were tailored to the dataset being used, reflecting the common sources of location error for each dataset. In previous studies, the locations of sensors used to collect spatio-temporal data are often recorded using GPS devices. In literature, the error in location recorded by GPS devices is shown to be, or be overbounded by, bivariate normal distributions independent in the  $x$  and  $y$  directions, with a mean value of 0 [63, 102]. In [102], an experiment with a stationary GPS logger that recorded 720 position estimates over 6 hours found the distribution of estimates was centred around the true location with 95% of instances occurring within 3 m of the true location. In [63], the maximum observed error in GPS location had a standard deviation of 1.258 m (approximately equal to 95% of instances occurring within 2.5 m of the true location). However, some spatial and spatio-temporal datasets use other referencing systems such as a grid-based referencing system [53]. In the transportation domain, locations within transport networks are often recorded as network link or edge locations. Therefore, studies on the robustness of spatial methods have to consider both the accuracy of location recording devices and the precision of the referencing system used to denote each location.

### 2.5.2 Errors in Timestamps of Instances

Similar analyses to those discussed for the spatial domain are used to analyse the sensitivity of methods to error in the timestamp recorded for each instance. For example, Delmelle *et al.* examined the effect of temporal error on Dengue Fever outbreak visualisations [40]. Using the known incubation period (4–10

days) of Dengue Fever, they estimated that the maximum delay between a person first experiencing symptoms and presenting symptoms to their doctor was 5 days. Therefore, to test the effect of temporal error, a temporal offset was applied to each instance, chosen from a Gaussian distribution with mean 0 and variance of 1.68 days (corresponding to a maximum value of 5 days using the *empirical rule* of 3 standard deviations). Similarly, Malizia’s burglary study used an empirical distribution of errors that was approximately exponential and reflected the estimated delay between the actual time of a burglary and the incident time recorded in the burglary dataset [87]. Values chosen from the distribution were absolute, and so were multiplied by a value chosen from a uniform distribution with limits  $[-1, 1]$ . Since Malizia used a single input dataset, 1000 perturbed datasets were created from the input dataset.

In cases where the inaccuracy of the recorded time for each instance cannot be estimated through empirical study, reasonable assumptions may be used instead. For example, when perturbing a dataset of hourly particulate sensor observations, Garcia-Menendez *et al.* estimated that the maximum lag between an event occurring and it being detected by particulate sensors was 3 hours [53]. This was a reasonable assumption gathered from domain knowledge rather than a value calculated from the data and, based on this assumption, they perturbed the instances by up to 4 hours from the original (clean) times they were recorded at. Similar studies may make use of domain knowledge of the sensors used to generate the dataset, accounting for common issues that affect sensors such as clock drift, temperature and pressure, and synchronisation issues with time servers. These issues may be (i) single offsets only, meaning that the clock for each sensor is offset by a fixed amount permanently and every instance at that sensor is perturbed by the same offset; (ii) cumulative, i.e. the error may compound over time, such as in the case of clock drift. In many systems, time synchronisation is performed using protocols such as the Network Time Protocol (NTP), which limits the maximum time between resynchronisations to 1024 seconds and the drift to less than 100 milliseconds [81]. Thus, when the temporal inaccuracy of a dataset cannot be estimated empirically, domain knowledge may be used to generate the perturbation distributions with which an analysis can be performed.

### 2.5.3 Noise in Feature Values

To study the impact of measurement error on spatio-temporal methods, analyses add noise to instances in the input dataset and measure the difference in methods’ output. In their analysis of the kNN imputation technique, Cheng *et al.* injected noise to a subset of instances by adding or subtracting 2 standard deviations of the features’ distribution to the selected instances [30].

In each test, they set  $n$  instances to have noise in each of 8 datasets, where  $n \in \{3\%, 6\%, 9\%, 12\%\}$ . This approach allowed them to analyse the impact of highly erroneous feature values on the kNN imputation technique for a range of values of  $k$ . In the weather domain, Bokde *et al.* added normally distributed noise to an air temperature dataset recorded at Nottingham Castle in England to test the impact on a missing data imputation technique [23]. The noise added was chosen from a normal distribution with mean 0 and standard deviations ranging from 0 to 8 degrees Fahrenheit. In the hydrological domain, Sivakumar *et al.* added normally distributed noise to a rainfall dataset corresponding to 0.04, 0.08 and 0.16 standard deviations of the original feature to rainfall data [117]. This tested the effect of increased amounts of noise on noise removal techniques.

In all of these studies, the distribution of noise chosen was a multiple of the standard deviation of the clean data, or the range of values observed. As with the choice of parameter values for perturbation analysis, the distribution of noise chosen may also be based on empirical studies or domain knowledge and reasonable assumptions about the type of noise inherent on the data collection process.

#### 2.5.4 Sources of Missing Instances

Missing instances in a dataset can affect the accuracy of methods such as imputation and prediction techniques. Data may be missing at random (MAR), in which random instances are missing from the dataset, or missing not at random (MNAR), in which the probability of an instance not existing in the dataset is dependent on the presence of nearby instances in space and time [30]. Instances may be missing at random due to events such as a temporary loss of power at the sensor or data corruption. Further, instances may be missing not at random due to censorship or the presence of conditions that made it inappropriate to sample the spatio-temporal process.

To study the impact of both types of missing data on imputation methods, Chen *et al.* removed increasing numbers of instances from a road traffic dataset [29]. They tested both MAR by removing between 10% and 50% of instances, and MNAR by selecting a random day and sensor and removing all instances recorded that day at that sensor, and repeating this process until the desired number of instances had been removed. For both types of missing data, the missing instances were imputed using a range of imputation techniques, and the error between the imputed values and removed instances reported.

A similar study in the transport domain was performed by Rodrigues *et al.* who removed 10%, 25%, 50% and 75% of the data at random, and Chuan *et al.* in the hydrology domain, who removed 5%, 10%, 15%, 20%, 25% and 30%

of the instances in a rainfall dataset to test to effect on imputation algorithms [34, 105]. In an analysis of the kNN technique in hydrology (where  $k = 10$ ), Lebecherel *et al.* tested the impact of removing all sensors within 10 km to 200 km of a withheld sensor being imputed, as well as removing between 10% and 90% of the sensors at random [77]. The distances chosen were selected by considering the mean distance between any sensor and its 10 nearest neighbours in the dataset. By analysing how the error incurred increased as the quantity of data removed increased, these studies allow us to infer the likely error incurred for other datasets.

## 2.6 Discussion

When analysing and linking spatio-temporal datasets, the volume of data to be processed can present a significant challenge for users. To aid this, data reduction techniques minimise the quantity of data to be processed. In particular, the reduction techniques presented in Section 2.2.5 that partition and model the data offer a clear advantage over selection and engineering/abstraction techniques, as well as data sketching techniques. These methods are shown to be effective at reducing the quantity of data stored while minimising the error incurred and retaining information about all features and instances.

However, at present, few methods exist for reducing spatio-temporal data by partitioning and modeling the data. A comprehensive survey of existing techniques for spatio-temporal data could not be found. Moreover, existing techniques exhibit several limitations. IDEALEM [78, 79] and the algorithm proposed by Pan *et al.* [97] remove entire partitions of instances, removing information that may be significant in later analysis. The ISABELA method overcomes this issue by storing a model for each partition, yet retrieving the data requires many instances be reconstructed and re-organised back into the spatial and temporal domains [75]. This adds significant computational overhead for data retrieval. The method proposed by Yang and Chen overcomes this issue, however it requires the user to select the number of blocks or partitions used for the dataset, and only considers the temporal domain of the data [142]. The use of fixed-size partitioning methods requires the user to select an appropriate partition size prior to reduction, ignoring the spatio-temporal dependencies in the data. This can reduce the storage efficiency of the reduced data and require more instances or coefficients be stored to capture the information present in the data.

In the data partitioning literature, the ST-DBSCAN algorithm may be used to overcome this issue [20]. By setting limits on both the dissimilarity and spatial/temporal distances between neighbouring instances that may belong to the same partition, the method considers both the feature values and spatio-

temporal proximity of instances when partitioning the data. However, this may still lead to high variance and spatio-temporal variability of the data, and not reduce the number of model coefficients required to store the reduced dataset. Furthermore, the user has to choose values for multiple parameters that may be unintuitive while the algorithm is sensitive to the values chosen.

Therefore, a new method is required that overcomes these issues and fulfils four requirements. First, it must use the variability of the data in space and time to partition the instances. Second, the method must capture information about all instances and features, using modelling techniques that limit the quantity of data used to capture the data. Third, the method must allow data to be retrieved without requiring multiple instances be imputed and processed. Finally, the method must permit some analysis using the reduced data. In Chapter 4, a novel method is presented that fulfils these requirements.

## Chapter 3

# Datasets

Spatio-temporal data can be captured from the physical world or generated through simulation. Data captured in the physical world may include environmental data collected from stationary sensors, retail data collected from stores, and telematics data collected from moving vehicles. Data generated through simulation may include computational fluid dynamics, climate or hydrology data. In both cases, each instance is referenced by the location in the spatial domain and time in the temporal domain at which it was recorded. Furthermore, the feature values of instances that are near to each other in space and time are more alike than those that are further apart.

For the spatial domain, one of many coordinate systems, or *spatial referencing systems* (SRSs), can be used. Spatial referencing systems define the coordinate space of the spatial domain used to locate instances in space, as well as specific map projections that convert coordinates in one SRS into coordinates in other SRSs. Different SRSs are used to reflect the topography of interest in different contexts. For example, the World Geodetic System (WGS 84) represents the world as a 2-dimensional ellipsoid and measures locations as degrees north and east of the International Earth Rotation and Reference Systems Service (IERS) Reference Meridian [49]. The distance between two locations in WGS 84 is calculated using an elliptical distance equation. However, in the transportation domain, Linear Referencing Systems (LRSs) are commonly used. Locations within linear referencing systems are described by their distance from a fixed origin along a path, such as a road or railway. Using a LRS rather than a 2-dimensional SRS better reflects the real distance required to travel between two locations within the bounds of a transport network, which can be longer than the distance travelled within the unbounded 2-dimensional coordinate system of WGS 84.

In some scenarios, the processes captured in spatio-temporal datasets are unconstrained in both the spatial and temporal domains. That is, the processes are continuous in space and time, and values can be imputed or estimated for



any point in the spatial and temporal domains. For example, air temperature can be recorded and imputed anywhere on Earth, although different recording apparatus may be required for different areas (e.g. on land versus at sea). However, in other scenarios the process may be confined, with bounds on where and when instances can be recorded or imputed. For example, water temperature can only be recorded and imputed in areas where water exists, and retail sales data may only be captured where retail stores exist. Thus, it is improper to impute water temperatures in dry areas and sales data where stores do not exist. Similarly, we can say motorway road traffic is confined to motorways and therefore cannot impute traffic count values for locations off-road. Such bounds are specific to the process captured in the data, although they may be captured within the spatial referencing system used (e.g. traffic locations may be described using a LRS rather than 2-dimensional SRS).

As well as describing the spatial referencing system used and the spatio-temporal bounds on the process, spatio-temporal datasets can be described by the distribution of instances in space and time, and by metrics that characterise the dataset’s feature values. First, understanding the dispersion of instances in space and time allows us to select appropriate methods for analysing and imputing instances in the data. Furthermore, understanding the dispersion of instances allows us to understand sampling error and bias in the data. Second, characterising the features in the dataset allows us to understand how the process evolved over space and time. The metrics used to describe spatio-temporal datasets in this thesis are outlined in Section 3.1, and the datasets used in this thesis are described in Section 3.2.

### 3.1 Characterising Spatio-Temporal Data

Given a spatio-temporal dataset  $D$ , the spatial domain of  $D$  can be denoted  $\mathbb{S} \subset \mathbb{R}^{\mathcal{D}}$ , where  $\mathcal{D}$  is the number of dimensions of the SRS used. Furthermore, the temporal domain can be denoted  $\mathbb{T} \subset \mathbb{R}$ , and an individual instance in  $D$  recorded at location  $s \in \mathbb{S}$  at time  $t \in \mathbb{T}$  denoted  $d_{s,t}$ . In this thesis, we focus on datasets generated by fixed sensors, meaning that the location of a sensor does not change over time. Furthermore, it is assumed that a maximum of one sensor can exist at any location  $s \in \mathbb{S}$ . Thus, the subset of sensor locations is denoted  $S \subset \mathbb{S}$ , and the subset of times at which instances are recorded is denoted  $T \subset \mathbb{T}$ . In this section, we describe each of the metrics considered in this thesis for describing spatio-temporal datasets.

### 3.1.1 Coverage of Space and Time

Instances in a spatio-temporal dataset may represent instantaneous samples of the spatio-temporal process, aggregated values of multiple readings, or represent the process over a spatial area or time period. The dispersion of these instances in space and time can be visualised using maps and timelines. Furthermore, the distribution of the distances between instances can be used to quantise this dispersion, as well as the rate of missing instances. These measures are discussed further in this section.

#### Point and Field Samples

Each instance in a spatio-temporal dataset may be an instantaneous sample of the process at a given location and time. Datasets of these instantaneous samples are commonly referred to as *object* [38] or *point datasets* [76]. That is, each instance represents the status of the process at the exact time and location recorded. Alternatively, each instance may refer to a period of time and/or an area of space, and such datasets are referred to as *field datasets*. For example, an instance may refer to a particular object that has a large spatial area such as a building or city, or be an aggregate of multiple samples collected in the defined area or time period.

The denotation of a dataset as a point or field dataset affects how its instances are analysed and processed. For example, the distance between two instances in a field dataset could be defined as the smallest distance between any two points on their perimeters, the largest such distance, or the distance between their mid-points. In this thesis, we focus on point datasets and do not consider field datasets.

#### Dispersion in Space and Time

Instances in spatio-temporal datasets may be regularly dispersed in space and time. That is, the distance between any instance and its nearest neighbour is constant, and this distance is the same for every instance (assuming no missing instances). For such datasets, the term *spatial resolution* is used to describe the distance between each instance and its nearest neighbour. However, instances are often not regularly spaced for multiple reasons. In the spatial domain, sensors may be irregularly spaced for one or two broadly defined reasons [51]. First, the spatio-temporal process may be more variable in some locations than others. Therefore, sensors that are used to sample the process may not be uniformly distributed in order to capture as much of the process' variability as possible. Second, the topography of the spatial field may prevent sensors being positioned in some locations, as may the availability of necessary utilities such as power and communications. When sensors are not uniformly dispersed, the

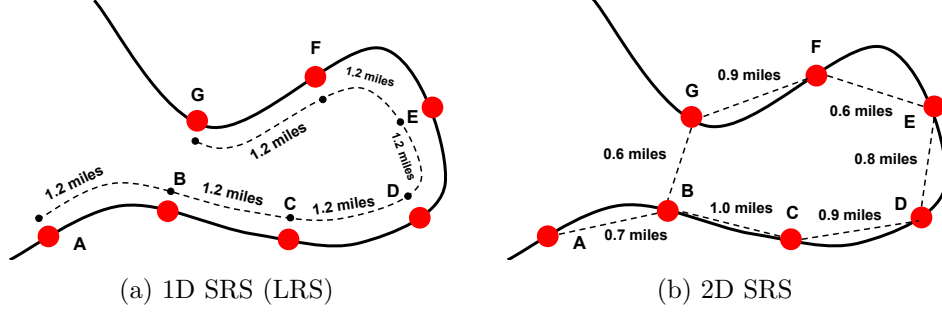


Figure 3.1: Nearest neighbours in a linear referencing system versus 2-dimensional SRS.

distance between each sensor and its nearest neighbour forms a distribution:

$$\{d \mid \forall s_i \in S : d = \arg \min_{s_j} \delta_S(s_i, s_j)\} \quad (3.1)$$

where  $s_i, s_j \in S$  and  $\delta_S(s_i, s_j)$  is a measure of spatial distance between the sensors  $s_i$  and  $s_j$ , and  $s_i \neq s_j$ . This distribution of distances can be calculated for both point and field datasets with appropriate choices of distance function.

It is important to note the significance of the SRS used when describing the dispersion of sensors in space and time. Consider the set of sensors depicted on the road shown in Figure 3.1. In Figure 3.1a, a linear spatial system is used and the sensors shown to be regularly dispersed with a resolution of 1.2 miles. However, in Figure 3.1b, a 2-dimensional SRS is used with the distances between nearest neighbours described as the sequence (0.7, 0.6, 0.9, 0.8, 0.6, 0.6, 0.6).

A similar process can also be used to describe the dispersion of instances in time. When the distance in time between consecutive instances at a sensor is constant, the dispersion of instances is referred to as the temporal resolution of the dataset. Furthermore, the dataset may be synchronous in time, meaning every sensor records an instance at the same time, or asynchronous. In the latter case each sensor records at regular intervals, although two sensors may record at different times. When sensors do not record at regular intervals, the temporal distance between instances at a sensor at location  $s_i$  is described by the distribution:

$$\{d \mid \forall t_j \in T_{s_i} : d = \arg \min_{t_k} \delta_T(t_j, t_k)\} \quad (3.2)$$

where  $T_{s_i}$  is the set of times at which the sensor at  $s_i$  recorded an instance,  $t_j, t_k \in T_{s_i}$ , and  $\delta_T(t_j, t_k)$  is a measure of temporal distance between  $t_j$  and  $t_k$ .

### Missing Instances

When a sensor is expected to record an instance at a given time but fails, that instance is said to be *missing*. As described in Section 2.5.4, instances may be missing at random or missing not at random. Furthermore, datasets may exhibit a *staircase effect*, in which sensors are brought online or taken permanently offline throughout the duration of time covered by the dataset. For a given sensor at location  $s_i \in S$ , the percentage of present instances for  $s_i$  is given by the ratio:

$$\text{Present}(s_i) = \frac{|s_i|}{|T_{s_i}|} \quad (3.3)$$

where  $|s_i|$  is the number of instances recorded at  $s_i$  and  $|T_{s_i}|$  is the number of discrete times that the sensor was expected to record an instance. A similar ratio for a dataset can be calculated as the ratio between the number of instances present divided by the number of sensors and time intervals possible:  $|D| / (|S| \times |T|)$ .

Missing instances may be highly distributed in time or form periods of consecutive missing instances — we can think of these informally as *periods of silence*. Given  $T_{s_i}$  is the ordered set of times the sensor at  $s_i$  is expected to record at, and  $t \in T_{s_i}$  is a timestamp in the ordered set, we can define the function  $I$  to indicate whether the sensor recorded an instance at time  $t$ :

$$I_D(s_i, t) = \begin{cases} 1, & \text{if } d_{s_i, t} \in D, \\ 0, & \text{otherwise} \end{cases}$$

Then, the set of temporal periods for which the sensor at  $s_i$  is missing instances is defined as:

$$\{t_l - t_j \mid \forall k \text{ s.t. } j < k < l, t_j, t_k, t_l \in T_{s_i} : \neg I_D(s_i, t_k) \wedge I_D(s_i, t_j) \wedge I_D(s_i, t_l)\} \quad (3.4)$$

that is, the set of time periods for which no instance was recorded within the time period but for which an instance was recorded before and afterwards. Finally, the dispersion of these temporal periods throughout the time series for the sensor at location  $s_i$  is defined as:

$$\{t_l - t_j \mid \forall k \text{ s.t. } j < k < l, t_j, t_k, t_l \in T_{s_i} : I_D(s_i, t_j) \wedge I_D(s_i, t_k) \wedge I_D(s_i, t_l) \wedge \neg I_D(s_i, t_{j-1}) \wedge \neg I_D(s_i, t_{l+1})\} \quad (3.5)$$

By understanding the distribution of lengths of time for which each sensor experienced a period of silence, and the distribution of lengths of time for

which each sensor recorded instances, appropriate methods for handling missing instances can be chosen.

### 3.1.2 Feature Values

The features of a spatio-temporal dataset record the values of one or more properties of the process being sampled. For example, a weather dataset may contain the features *temperature* and *humidity*. In spatio-temporal data, both the feature values and how those values vary over space and time are of interest. In literature, the features of spatio-temporal datasets are occasionally described using Shannon information theoretic measures. Examples can be found that use the entropy of the data to describe the range of instances observed, such as in the domains of hydrology [6, 124] and urban analytics [26, 32]. However, these ignore the spatial and temporal dependencies present in the data. Though some informal attempts have been made to account for spatio-temporal dependency in information theoretic measures [19, 35, 94], there is little consensus on how to do so [148]. Therefore, this thesis focuses on the use of descriptive statistics to describe the set of feature values present in each dataset, and techniques such as variograms to characterise the spatio-temporal dependencies present.

### Descriptive Statistics

Descriptive statistics quantitatively describe and summarise the values observed for each feature. Descriptive statistics are fundamental methods for describing the feature values in a dataset, and can be broken into distinct categories:

1. **Measures of central tendency** describe the mid-point of the feature. These include the arithmetic mean, median and, for discrete features, the mode.
2. **Measures of dispersion** describe how widespread the values of the feature values are. These include the range of the feature (the minimum and maximum values) and the variance.
3. **Measures of symmetry and tailedness** describe how the feature values are dispersed around the measures of central tendency. These include skew and kurtosis.

### Variation over Space and Time

As well as understanding the dispersion of instances over space and time, it is important to understand how the feature values of the instances vary over space and time [10]. To do so, an *empirical variogram* is used, which visualises the difference in feature value between pairs of instances in space or time (a separate

variogram is plotted for the spatial and temporal domains). The distance, or *lag*, between each combination of instances is calculated alongside a function of the difference between their feature values, and these two values plotted against each other. In practice, a discrete set of distance bins  $s \pm \Delta$  is used, where  $s$  is any spatial location in the dataset and  $\Delta$  is a spatial distance range, rather than the precise distance between instances. Furthermore, isotropic conditions are assumed, meaning the difference between feature values is a function of the distance between the instances, rather than a function of any other variables.

For a single spatial lag  $s \pm \Delta$ , the empirical spatial variogram  $\hat{\gamma}_S$  for  $s \pm \Delta$  is:

$$\hat{\gamma}_S(s \pm \Delta) = \frac{1}{2|N_S(s \pm \Delta)|} \sum_{(s_i, s_j) \in N_S(s \pm \Delta)} \sum_{t_k \in T} |d_{s_i, t_k} - d_{s_j, t_k}|^2 \quad (3.6)$$

where  $N_S(s \pm \Delta)$  yields the set of sensor pairs that are approximately  $\Delta$  spatial distance apart, and  $T$  is the set of discrete time intervals present in  $D$ . When either  $d_{s_i, t_k}$  or  $d_{s_j, t_k}$  is missing, that pair of instances is not considered in the calculation of  $\hat{\gamma}_S(s \pm \Delta)$ .

For a single temporal lag  $t \pm \Delta$ , the empirical spatial variogram  $\hat{\gamma}_T$  for  $t \pm \Delta$  is:

$$\hat{\gamma}_T(t \pm \Delta) = \frac{1}{2|N_T(t \pm \Delta)|} \sum_{(t_i, t_j) \in N_T(t \pm \Delta)} \sum_{s_k \in S} |d_{s_k, t_i} - d_{s_k, t_j}|^2 \quad (3.7)$$

As well as the empirical variogram, which visualises the spatial autocorrelation in the data, an adaptation of *Moran's Index* (Moran's I) for spatio-temporal data can be used to quantify the degree of autocorrelation between instances [141]. Moran's I is an extension of Pearson's Correlation Coefficient (PMCC, denoted  $r$ ). While PMCC measures the correlation between two variables or features, Moran's I quantifies the degree of correlation within the same feature for given spatial distance. Values of Moran's I that are close to +1 indicate a near-perfect clustering of similar instances in space, while a value of 0 indicates truly random dispersion of instances in space and a value close to -1 indicates near-perfect dispersion of similar instances in space (or perfect clustering of dissimilar instances). The statistic can be calculated using the equation:

$$I = \frac{1}{\mathcal{W}\hat{\sigma}^2} \sum_{s_i \in S} \sum_{s_j \in S} \sum_{t_k \in T} \sum_{t_l \in T} w_{i,j,k,l} (d_{s_i, t_k} - \bar{D})(d_{s_j, t_l} - \bar{D}) \quad (3.8)$$

where  $\mathcal{W}$  is the sum of weights,

$$\mathcal{W} = \sum_{s_i \in S} \sum_{s_j \in S} \sum_{t_k \in T} \sum_{t_l \in T} w_{i,j,k,l} \quad (3.9)$$

and the sample mean  $\bar{D}$  is given by  $\sum_{s \in S} \sum_{t \in T} d_{s,t} / |D|$ , and  $\hat{\sigma}^2 = \sum_{s \in S} \sum_{t \in T} (d_{s,t} - \bar{D})^2 / |D|$ . The set of weights  $\mathcal{W}$  between instances is determined by the spatio-temporal distance between the instances with  $w_{i,i,k,k} = 0$ . A discussion on combining spatial and temporal distance metrics into a single metric is presented in Section 5.1.1.

Moran's I can be used to quantify the correlation between all instances recorded at the same time, giving the Global Moran's Index. However, by setting the weight of all non-adjacent instances to 0, the Local Moran's Index for adjacent sensors can be computed giving the correlation between sensors that are adjacent in space. Here, we define two instances as being adjacent if they were recorded at the same sensor at consecutive times, or recorded at the same time at spatially adjacent sensors. A more in-depth discussion on spatio-temporal adjacency is presented in Section 4.2.1.

## 3.2 Datasets Used in this Thesis

The focus of this thesis is data collected and commonly used within the transportation domain. Three datasets are used: road traffic, air temperature and rainfall. For the road traffic dataset, 7 highways (motorways and A-roads) are used, and 12 1-month samples collected in 2017 are used for each highway. For the air temperature and rainfall datasets, 12 1-month samples collected in 2017 are also used. All of the datasets are point datasets, meaning that each instance is defined by a singular point in space and time. However, each instance in the traffic dataset was calculated over a 15 minute time interval ending at the time the instance was recorded at a point location, and each instance in the rainfall dataset was calculated over a 1 hour interval ending at the time the instance was recorded<sup>1</sup>. These datasets are used in the analysis presented in Chapters 4, 5 and 6. Each of the three datasets are described in this section and are summarised in Table 3.1.

### 3.2.1 Road Traffic

A set of 84 samples of the WebTRIS traffic dataset [47] are used in this thesis. Each sample is a month-long survey of traffic counting sensors taken from 7 highways (motorways and A-roads) in England between January and December 2017. Each sample is synchronous, with instances recorded at 15 minute intervals. Furthermore, the traffic dataset contains 6 real-valued features: *count of vehicles of length 0 m to 5.2 m*, *count of vehicles of length 5.21 m to 6.6 m*, *count of vehicles of length 6.61 m to 11.6 m*, *count of vehicles of length 11.61 m or greater*, *total count of vehicles* and *average speed (MPH)*. Each

---

<sup>1</sup>Despite this, these datasets are still point datasets as the instances are defined by a single point in space and time, i.e. the period end time.

Table 3.1: Characteristics of the datasets used in this thesis.

	Spatial Variance	Temporal Variance	Discontinuities in Space	Discontinuities in Time
Traffic	Medium	High	Low variation between sensors on main carriageway, high variation between main carriageway and slip road sensors	Road traffic collisions introduced discontinuities in time
Air Temp.	Low	Low	Nearby sensors recorded similar feature values	Small variations between adjacent time intervals
Rainfall	Changes over time	Low	Rainfall events localised to groups of nearby sensors at the same time	Varies substantially over time as rainfall events occur

sample exhibits daily trends as well as weekly trends, with public holidays demonstrating similar patterns as weekends in some cases. Sensors on slip roads (entries and exits) are interspersed amongst main carriageway sensors, and exhibit much lower traffic counts compared to the main carriageway sensors.

An initial investigation was carried out to select the 7 highways used in this thesis. The traffic characteristics of each highway in England was considered and discussed with traffic data analysts. Each of the highways were characterised by their daily patterns and weekly/yearly trends, the proportions of different vehicle sizes and speeds observed, their geographic location and the quantity of traffic overall. From this investigation, the A30, A66 and A69 arterial roads were selected, as well as the M1, M11, M20 and M56 motorways.

The samples used from this dataset contain between 46,053 and 311,148 instances each, and range from 3.3 MB to 23.6 MB in size. Table 3.2 shows the percentage of instances present in each sample of data, and Table 3.3 shows the number of sensors in each dataset. As shown, the M20 exhibits the lowest percentage of present instances in any one month (46.2%), and the A69 and M1 exhibits the highest percentage of present instances in any one month (99.9%). Overall, the M20 exhibits the lowest mean percentage of present instances (83.2%), and the A69 exhibits the highest mean percentage of present instances (90.5%).

A map of the road sensor locations can be seen in Figure 3.2a. Each of the highways exhibits a different range of distances between each sensor and its nearest neighbour, as shown in Figure 3.3a. Note that the majority of distances are less than 7 miles, so the range of the figure is limited to 7 miles.



Table 3.2: Percentage of instances present in the traffic datasets. Values in bold are the minimum and maximum mean values across all roads and months.

	A30	A66	A69	M1	M11	M20	M56	<i>Mean</i>
Jan	81.6	77.2	77.7	86.2	79.5	83.5	86.0	81.7
Feb	93.6	86.9	96.4	94.4	86.1	94.0	93.5	92.2
Mar	90.7	88.9	93.4	88.4	83.7	86.4	81.1	87.5
Apr	95.6	96.9	99.9	99.8	96.2	89.5	99.8	<b>96.8</b>
May	62.8	60.7	61.9	66.7	64.8	62.9	66.3	<b>63.7</b>
Jun	94.1	90.7	95.2	99.8	97.0	85.5	96.8	94.2
Jul	92.6	85.6	93.9	51.1	51.5	46.2	50.2	67.3
Aug	91.9	82.6	94.2	96.0	86.4	85.3	94.3	90.1
Sep	93.1	87.6	93.3	99.9	96.9	89.6	98.3	94.1
Oct	93.9	95.8	96.9	99.1	89.8	92.5	98.6	95.2
Nov	85.1	86.4	87.6	91.5	88.3	89.4	87.8	88.0
Dec	99.2	95.2	95.0	94.1	94.3	93.1	96.3	95.3
<i>Mean</i>	89.5	86.2	<b>90.5</b>	88.9	84.6	<b>83.2</b>	87.4	87.2

Table 3.3: Number of sensors in each dataset, calculated across 12 1-month samples.

	A30	A66	A69	M1	M11	M20	M56	Air Temp.	Rainfall
Min	76	46	23	48	48	103	55	407	258
Mean	78	48	25	49	52	111	56	447	259
Max	79	49	25	50	60	117	62	461	260

The mean nearest neighbour distance between sensors ranges from 0.01 miles for the M20 to 1.21 miles for the A30, with the maximum distance between any sensor and its nearest neighbour being 26.1 miles on the A30. Furthermore, the total duration for periods of missing instances can be seen in Figure 3.3d. In many cases, only 1 instance is missing, giving a period of inactivity of 30 minutes. However, for many highways, a significant number of periods lasted for 24 hours and 72 hours.

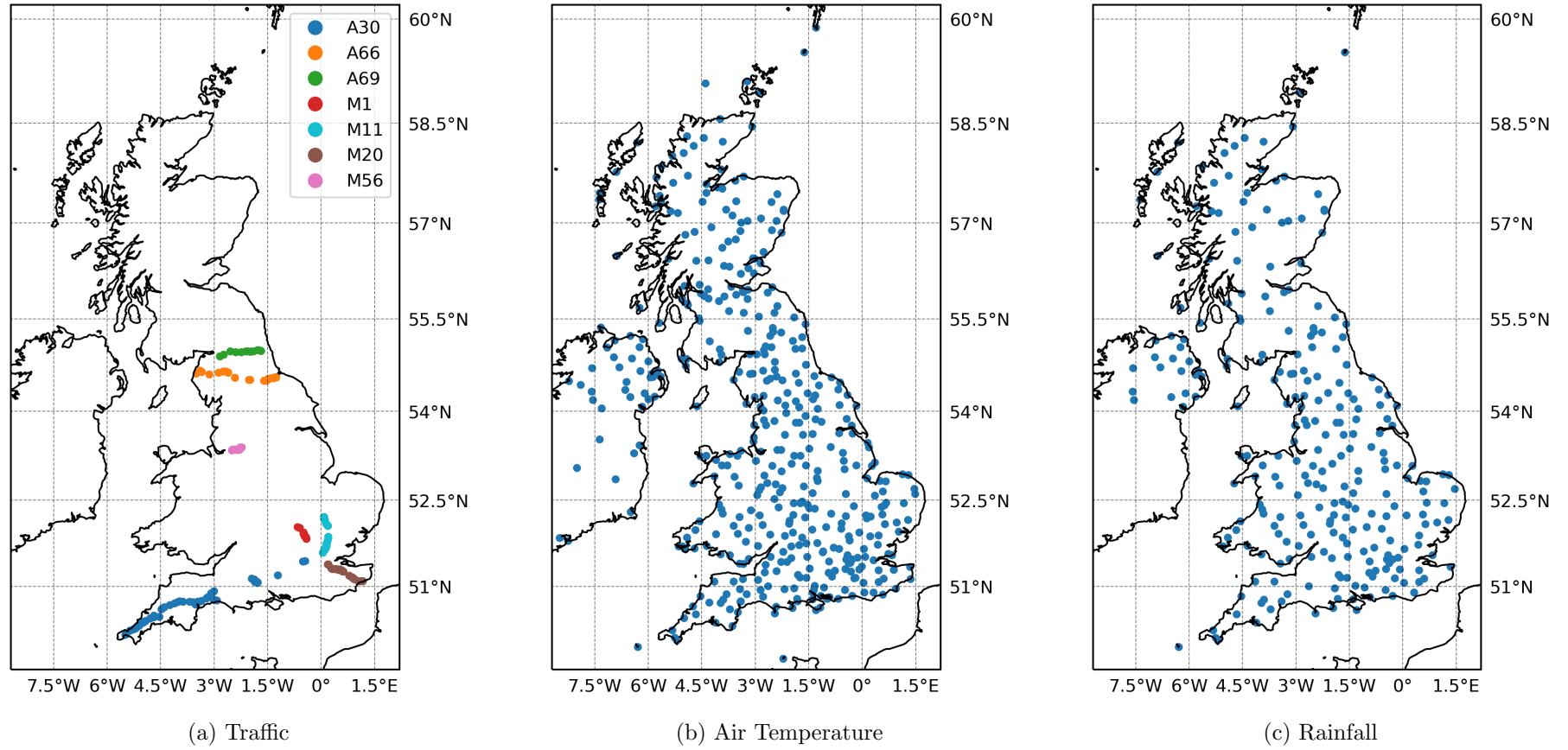
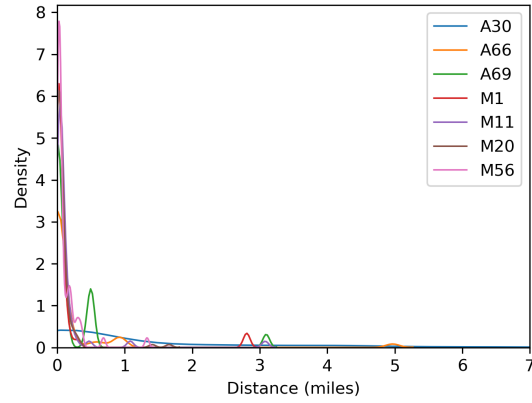
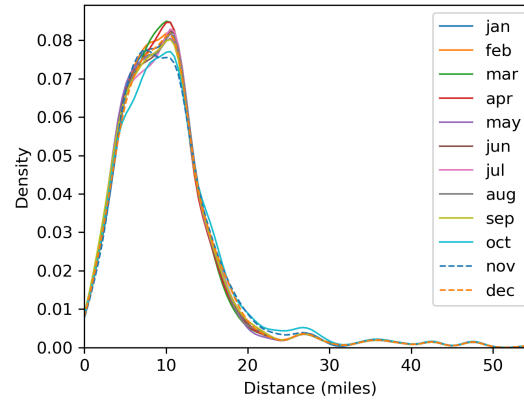


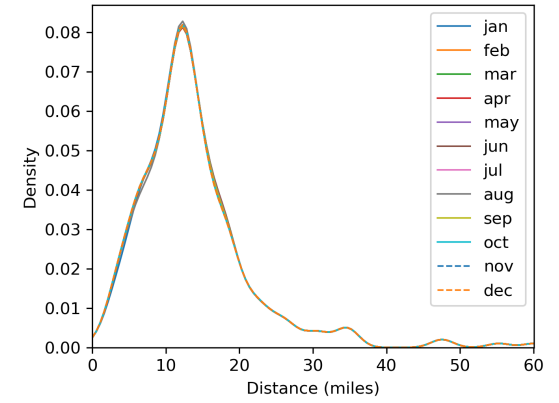
Figure 3.2: Locations of the traffic, air temperature and rainfall sensors used in this thesis.



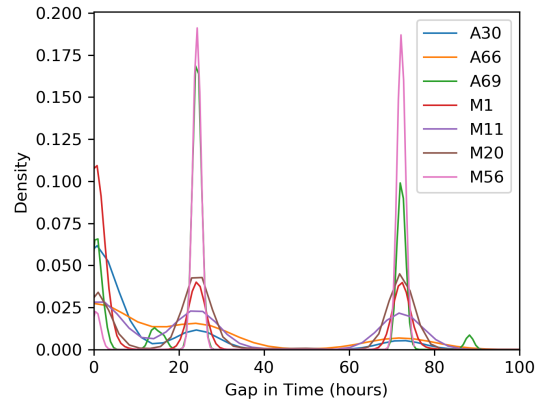
(a) Distances between Sensors  
**Traffic (Jan)**



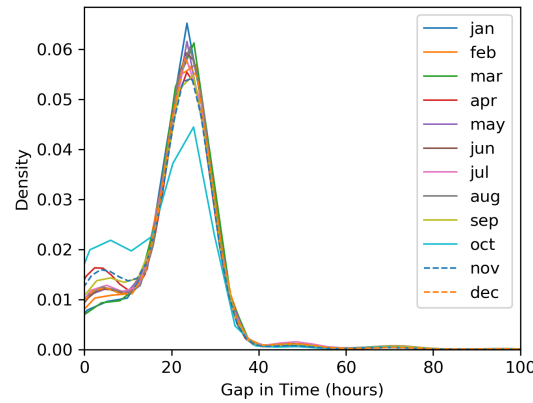
(b) Distances between Sensors  
**Air Temperature**



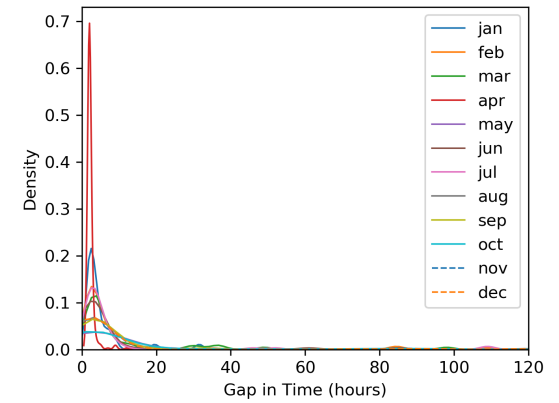
(c) Distances between Sensors  
**Rainfall**



(d) Periods of Inactivity  
**Traffic (Jan)**



(e) Periods of Inactivity  
**Air Temperature**



(f) Periods of Inactivity  
**Rainfall**

Figure 3.3: Distance between sensors and lengths of periods of inactivity for the datasets used in this thesis.

## Feature Values

Each of the highways exhibit different distributions for each of the 6 traffic features. This is demonstrated in Figures 3.4a and 3.4b which show the density plots of the total vehicle count and average speed respectively. These figures show the density of the two features for July 2017, although the density plots for each month are approximately the same. Each highway exhibits different quantities of traffic flow and differing average speed. As shown in Figure 3.4c, the proportion of vehicle sizes counted varies between highways, although these proportions were approximately the same for each of the 12 months of data.

For all samples of the traffic dataset, the variance of all features in space is significantly higher than the variance in time for instances that are close together. For example, as shown in Figure 3.5a, the variogram value (Equations 3.6 and 3.7) of the total volume feature is significantly lower for instances that are close in time than for instances that are close in space. This is mainly attributed to the presence of road works and the difference between slip roads (entries on and off the main motorway) and the main motorway. Such conditions cause some sensors that are close in space to record significantly different values for all of the dataset’s features. However, sensors on the main carriageway that are close in space exhibit similar feature values at the same time, meaning there is still high spatial autocorrelation (low variation) between main carriageway sensors despite there being low spatial autocorrelation between slip road sensors, although this cannot be distinguished from a spatial variogram.

### 3.2.2 Air Temperature

A set of 12 samples of air temperature data were collected from the Met Office Integrated Data Archive System (MIDAS) Land and Marine Surface Stations Dataset [95] in the United Kingdom (UK) and Ireland. Each sample consists of a month-long survey of air temperature sensors in England between January and December 2017. Furthermore, each sample is synchronous, with instances recorded at 1-hour intervals. The dataset consists of a single real-valued feature, *temperature (degrees Celsius, °C)*, and each sample exhibits daily trends. Unlike the traffic dataset, the air temperature dataset is unaffected by public holidays, although each day exhibits an increased temperature during the daytime and decreased temperature at night.

The samples used from this dataset contain between 230,291 and 254,672 instances each, and range from 10.4 MB to 11.7 MB in size. Table 3.4 shows the percentage of instances present in each sample of data, with a mean of 447 sensors present in each sample (minimum 407, maximum 461). As shown, March exhibits the lowest percentage of present instances (74.0%), while October exhibits the highest percentage of present instances (83.1%).

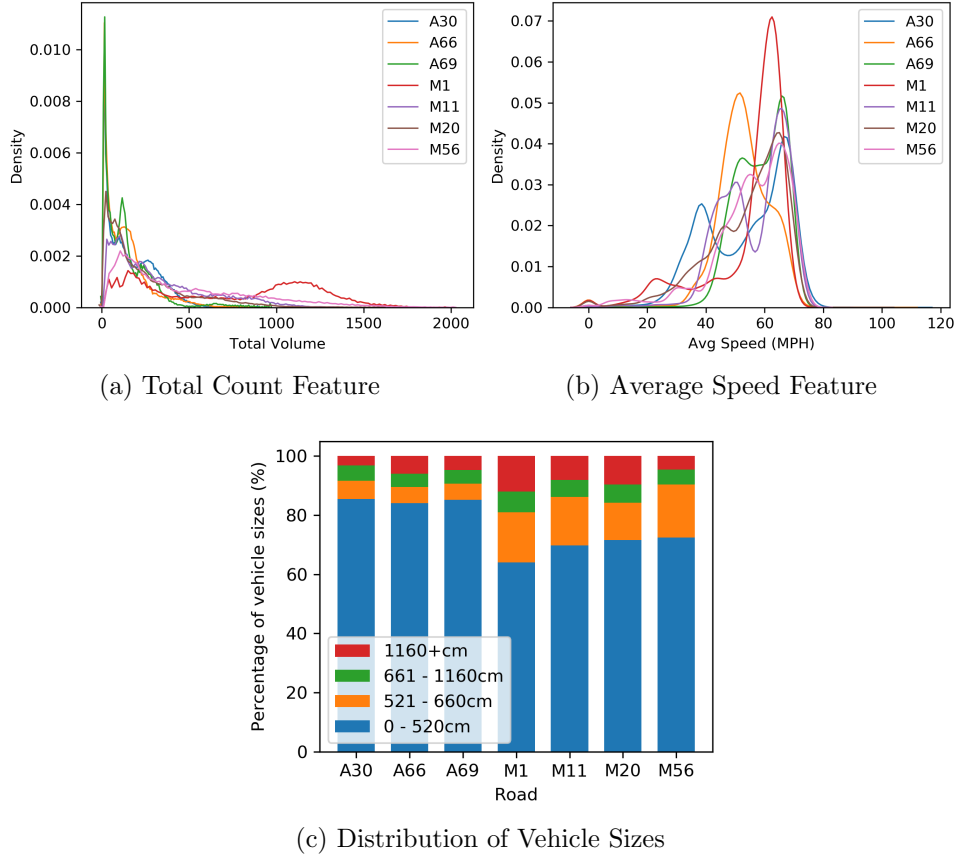


Figure 3.4: Values of the total vehicle count and average speed features, as well as the distribution of vehicle sizes, for July 2017.

A map of the road sensor locations can be seen in Figure 3.2b. Furthermore, a histogram of the distance between each sensor and its nearest neighbour can be seen in 3.3b, while the total duration for periods of missing instances can be seen in Figure 3.3e. The mean nearest neighbour distance in the air temperature dataset is 10.3 miles, with a maximum distance of 55.3 miles. In the temporal domain, some periods of inactivity lasted less than 24 hours, although a significant number of periods lasted exactly 24 hours.

### Feature Values

Each of the 12 air temperature samples exhibits a different mean and range of values for the temperature feature. The range of values is smaller for summer months (April to October) as shown in Figure 3.6a. However, the feature follows an approximately normal distribution, with December and February showing bimodal distributions. The mean value for each month ranges from 4.39 (January) to 15.47 (July), and the standard deviation for each month ranges from 3.13 (September) to 4.36 (May).

The feature values of the temperature feature vary more in time than space,

as shown in Figure 3.5b. This figure shows the variogram value over space and time for all 12 samples of air temperature data, although approximately the same distribution is seen for each of the individual samples. As shown, large areas of space experienced similar feature values at the same time, although the difference between instances recorded more than 3 hours apart at any single sensor was higher and increased as the time period between the two instances increased.

### 3.2.3 Rainfall

A set of 12 samples of rainfall data were collected from the MIDAS Land and Marine Surface Stations Dataset [95] in the UK. Each sample consists of a month-long survey of rainfall sensors in England between January and December 2017. Furthermore, each sample is synchronous, with instances recorded at 1-hour intervals. The dataset consists of a single real-valued feature, *total hourly precipitation (mm)*. Unlike the traffic and air temperature datasets, the rainfall dataset does not exhibit daily or weekly trends.

The samples used from this dataset contain between 172,908 and 191,766 instances each, and range from 8.2 MB to 9.2 MB in size. Table 3.4 shows the percentage of instances present in each sample of data, with a mean of 259 sensors present in each sample (minimum 258, maximum 260). As shown, March exhibits the lowest percentage of present instances (98.6%), while June exhibits the highest percentage of present instances (99.8%). A map of the road sensor locations can be seen in Figure 3.2c. Furthermore, a histogram of the distance between each sensor and its nearest neighbour can be seen in 3.3c, while the total duration for periods of missing instances can be seen in Figure 3.3f. The mean nearest neighbour distance in the rainfall dataset is 13.8 miles, with a maximum distance of 60.1 miles. In the temporal domain, most periods of inactivity lasted less than 10 hours, with almost no periods of inactivity lasting more than 24 hours.

### Feature Values

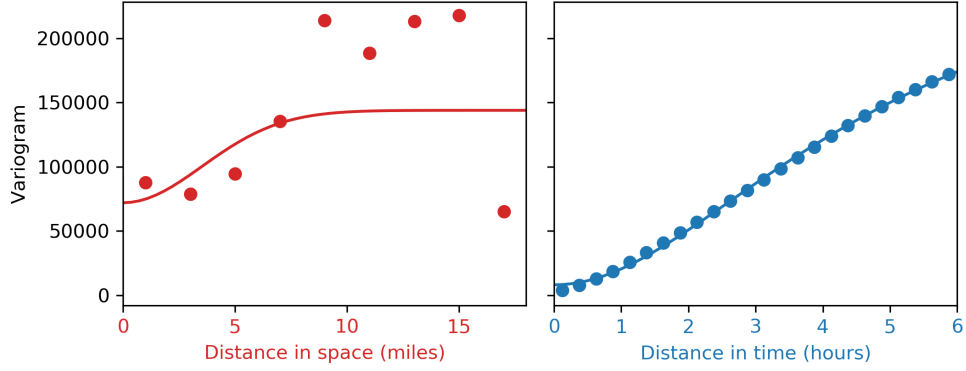
The rainfall dataset samples exhibit long-tail distributions with the majority of instances being 0 mm rainfall. Unexpectedly, July and August show an increased percentage of non-zero instances. The range of values observed for each month is shown in Figure 3.6b. The maximum value for each month ranges from 9.2 mm (February) to 47.8 mm (July), although only values up to 10 mm are shown in Figure 3.6b.

The feature values of the precipitation feature vary more in time than space, although the difference between time and space is smaller when compared to the temperature feature. The variogram for the precipitation feature is

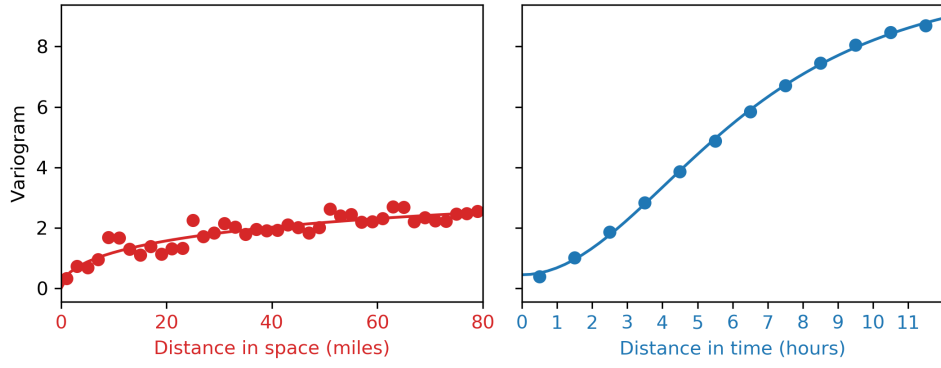
Table 3.4: Percentage of instances present in the air temperature and rainfall datasets.

	Air Temperature	Rainfall
Jan	74.9	99.5
Feb	74.7	99.0
Mar	74.0	98.6
Apr	75.0	99.4
May	75.2	98.9
Jun	75.2	99.8
Jul	76.4	99.2
Aug	76.1	99.2
Sep	75.7	99.1
Oct	83.1	99.0
Nov	79.2	99.6
Dec	75.8	99.3
<i>Mean</i>	76.2	99.2

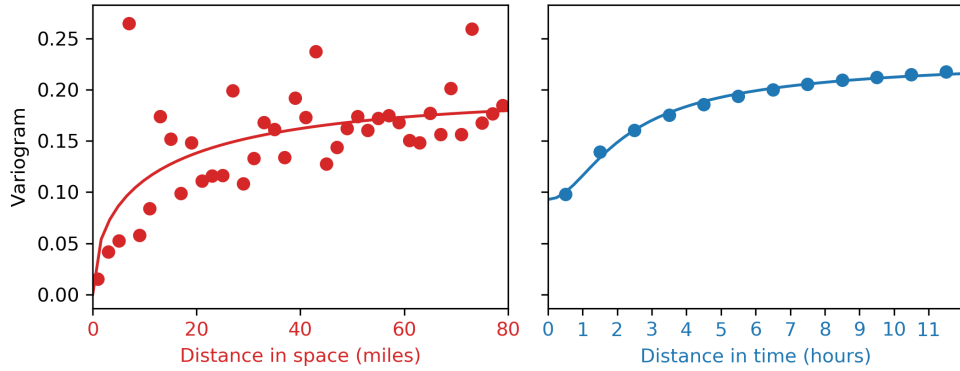
shown in Figure 3.5c. This figure shows the variogram value over space and time for all 12 samples of rainfall data. Approximately the same distribution is seen for each of the individual samples, although the difference in space is higher overall for July than for the other months with peak temperatures in the south of England being higher than in other months. However, in all months the variance in time is higher than the variance in space. Unlike the air temperature dataset, the difference between instances recorded 1 hour apart is significantly higher than instances recorded 10 miles apart.



(a) Traffic (Total Count feature for M1)



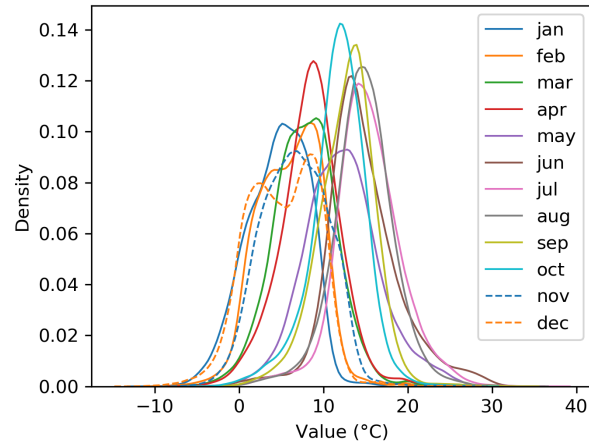
(b) Air Temperature



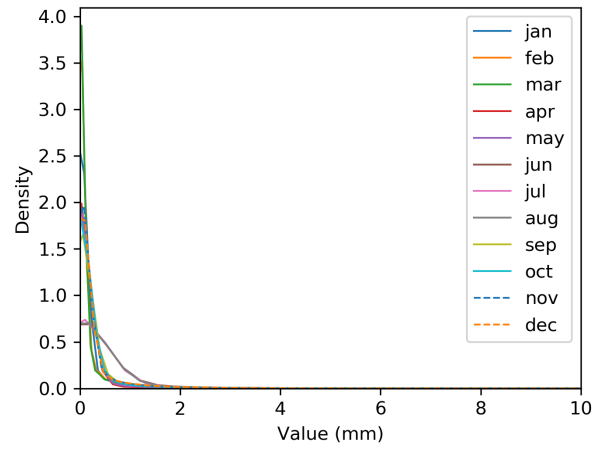
(c) Rainfall

Figure 3.5: Variograms for the traffic, air temperature and rainfall datasets. The variograms are calculated across all months for each of the three datasets, and the traffic variogram is shown for the total vehicle count of the M1 motorway.





(a) Air Temperature Feature



(b) Total Precipitation Feature

Figure 3.6: Values of the air temperature and total precipitation features, calculated over January to December 2017.

## Chapter 4

# Reducing by Partitioning and Modelling

To overcome the problems posed by increasing dataset sizes, several reduction algorithms have been introduced as discussed in Chapter 2. Given an input dataset, these methods transform the data into a reduced form referred to as the *reduced dataset*. The original instances can then be recovered by retrieving them from the reduced dataset, and some methods permit the imputation of unseen instances from the reduced form. However, as discussed in Section 2.6, these methods exhibit several shortcomings. First, methods that remove features and instances may reduce information from the data that is important in later analysis. To support many types of analysis, the reduced dataset should capture information about all instances and features in the data. Second, methods such as ISABELA require many instances to be imputed and processed to retrieve any single instance from the reduced dataset. This inefficiency can slow down the processing of the reduced dataset.

Therefore, the main contribution of this chapter is a new spatio-temporal reduction algorithm that overcomes the weaknesses of the methods presented in Chapter 2. We refer to this as the *k-Dimensional Spatio-Temporal Reduction* (*kD-STR*) algorithm. Within this contribution, a novel hierarchical partitioning technique is presented for spatio-temporal datasets which uses the variation of the feature values over space and time to partition the data in the spatial and temporal domains. A hierarchical partitioning allows *kD-STR* to choose the appropriate number of partitions given the users' preference for storage reduction. After fitting a model to each partition, *kD-STR* is able to store a reduced form of the dataset by storing the partition boundaries and model coefficients.

In this chapter, the *kD-STR* algorithm is introduced and analysed using the datasets presented in Chapter 3. In Section 4.1, the preliminaries of the algorithm are presented. In Section 4.2, *kD-STR* is detailed and, in Sections

4.2.1 and 4.2.2 respectively, the partitioning and modelling steps of the  $k$ D-STR algorithm are explained. Section 4.3 describes the experimental setup used to evaluate  $k$ D-STR, and Section 4.4 explains the results of this evaluation. In Section 4.5, the consequences of these results are discussed. Finally, Section 4.6 provides concluding remarks for this chapter.

## 4.1 Reducing Datasets to Partitions and Models

In this thesis, a spatio-temporal dataset  $D$  is defined as a set of instances generated by a set of synchronous or asynchronous sensors. Each instance in  $D$  is recorded at a location  $s \in S$ , where  $S$  is the subset of sensor locations in the continuous spatial domain  $\mathbb{S} \subset \mathbb{R}^{\mathcal{D}}$ , and at a time  $t \in T$ , where  $T$  is the subset of time intervals recorded at in the continuous temporal domain  $\mathbb{T} \subset \mathbb{R}$ . Each instance in  $D$  is therefore referenced using the notation  $d_{s,t}$ . Furthermore, the number of spatial and temporal dimensions is denoted  $k$ , where  $k = \mathcal{D} + 1$ . In this thesis, it is assumed that each sensor exists at a unique location and records at most one instance at any given time, thus only one instance can exist at a given location and time  $(s, t)$ .

Each instance  $d_{s,t}$  is a vector of values over the set of features  $F$ , i.e.  $d_{s,t} = \langle d_{s,t}^1, \dots, d_{s,t}^{|F|} \rangle$ . For example, the set of features in the traffic datasets used in this thesis includes *average speed* and *total vehicle count*. For generality we will consider real-valued features in this thesis. Therefore, the dataset is a mapping from the  $k$ -dimensional spatio-temporal space to the  $|F|$ -dimensional feature space,  $D : \mathbb{S} \times \mathbb{T} \rightarrow \mathbb{R}^{|F|}$ .

To reduce the dataset  $D$ , the aim of model-based reduction methods is to find a set  $P = \{p_1, \dots, p_{|P|}\}$  of non-overlapping partitions in the  $\mathbb{S} \times \mathbb{T}$  space. Each partition  $p_i \in P$  is defined by a bounding spatial polygon  $b_i$  in  $\mathbb{S}$ , a beginning time  $t_i^b$  and ending time  $t_i^e$ . The subset of instances in  $D$  that exist within the spatial and temporal bounds of  $p_i$  is denoted  $D_{p_i} = \{d_{s,t} \in D \mid \text{inside}(s, b_i), t_i^b \leq t \leq t_i^e\}$  and  $\langle b_i, t_i^b, t_i^e, D_{p_i} \rangle = p_i$ . Here, the function  $\text{inside}(s, p_i)$  is used to indicate that location  $s$  is within the bounding polygon  $b_i$  of partition  $p_i$ . Each partition must contain at least one instance, that is  $\forall p_i \in P : D_{p_i} \neq \emptyset$ , and every instance in  $D$  must belong to exactly one partition,  $\forall p_i, p_j \in P : p_i \cap p_j = \emptyset$  and  $\cup_{i=1}^{|P|} p_i \in P = D$ .

### 4.1.1 Partitioning and Modelling Spatio-Temporal Data

As discussed in Section 2.3, existing work found in literature uses a fixed-size partitioning scheme to partition a dataset in the spatial and temporal domains. This results in instances that are near to each other in space and time being placed in the same partition without considering their similarity in the feature

space. Consequently, the variance within each partition can be high, requiring a greater number of model coefficients to capture the variability of the data. Instead, it may be more beneficial to consider the similarity of instances in the feature space when partitioning the spatial and temporal domains.

Measures of similarity in the feature space are commonly used in clustering. These can be divided into inter-cluster metrics and intra-cluster metrics. Inter-cluster metrics separate instances by maximising the distance between clusters in the feature space, thereby maximising the dissimilarity between instances in different clusters. Given a distance metric  $\delta_F(d_{s,t}, d_{u,v})$  which measures the distance between two instances  $d_{s,t}, d_{u,v} \in D$  in the feature space, inter-cluster metrics include [93]:

- (a) the minimum (single) linkage criteria:  

$$\min\{\delta_F(d_{s,t}, d_{u,v}) : d_{s,t} \in D_{p_i}, d_{u,v} \in D_{p_j}\},$$
- (b) the maximum (complete) linkage criteria:  

$$\max\{\delta_F(d_{s,t}, d_{u,v}) : d_{s,t} \in D_{p_i}, d_{u,v} \in D_{p_j}\},$$
- (c) the unweighted average linkage criteria:  

$$\frac{1}{|p_i| \cdot |p_j|} \sum_{d_{s,t} \in D_{p_i}} \sum_{d_{u,v} \in D_{p_j}} \delta_F(d_{s,t}, d_{u,v}).$$

However, inter-cluster metrics only maximise the dissimilarity between clusters and do not consider the density of instances within each cluster, thereby increasing the variance within each cluster. In contrast, intra-cluster metrics can be used to minimise measures of dissimilarity within each cluster, such as variance. In particular, the Ward criterion joins the two clusters that minimise the overall increase in variance across all clusters [134]. For each pair of clusters, the Ward criterion measures the variance within each cluster as well as the variance within the proposed cluster if the those two clusters are joined. On each iteration, the pair of clusters that leads to the smallest increase in variance are joined. To measure the variance within each cluster, Ward uses the squared Euclidean distance between each instance within a cluster and the cluster's centroid in the feature space. Thus, Ward minimises:

$$\delta_{\text{Ward}}(c_i, c_j) = \sum_{d_{s,t} \in c_i \cup c_j} \delta_F(d_{s,t}, \bar{c}_{ij})^2 - \sum_{d_{s,t} \in c_i} \delta_F(d_{s,t}, \bar{c}_i)^2 - \sum_{d_{s,t} \in c_j} \delta_F(d_{s,t}, \bar{c}_j)^2 \quad (4.1)$$

where  $\bar{c}_i$  and  $\bar{c}_j$  are the centroids of clusters  $c_i$  and  $c_j$  respectively in the feature space, and  $\bar{c}_{ij}$  is the centroid of  $c_i \cup c_j$ .

Thus, by finding groups of similar instances in the feature space, and joining those instances into contiguous partitions in the spatio-temporal domain, the issue of fixed-size partitioning as is used in state-of-the-art methods can be overcome. The partitions are not restricted to regular shapes in the spatial

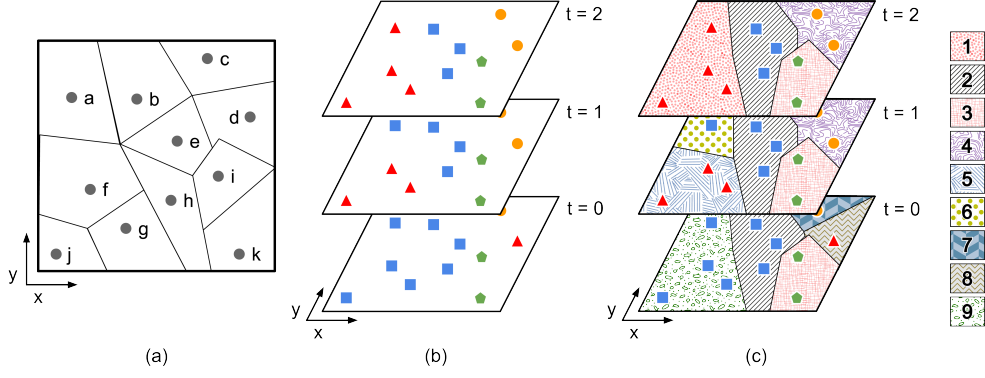


Figure 4.1: Partitioning of instances in the  $\mathbb{S} \times \mathbb{T}$  space. (a) Sensors in a spatio-temporal dataset are shown at their spatial locations with the space decomposed into Voronoi polygons. (b) Instances recorded at these sensors, coloured by their similarity. (c) The instances grouped into spatio-temporal partitions, where each partition is defined by the union of the Voronoi polygons of its constituent sensors, a start time and end time.

domain and may span multiple time periods. An example of the desired partitioning for a dataset can be seen in Figure 4.1(c). In (a), sensors from  $D$  are shown at their locations in the 2-dimensional spatial domain, which has been split into Voronoi polygons surrounding each sensor. In (b), instances recorded at the sensors have been colour coded by their similarity. For example, instances denoted by blue squares are more similar, or closer in the feature space, than to other instances denoted by red triangles, yellow circles and green pentagons. Finally, in (c), the instances from (b) have been grouped into 9 partitions, wherein the instances within each partition are more alike than the instances in their neighbouring partitions. Although the process illustrated in Figure 4.1 considers a 2-dimensional spatial domain, i.e.  $\mathcal{D} = 2$ , a similar partitioning may be found for any number of spatial dimensions.

After partitioning the instances of  $D$  in the spatio-temporal domain, a model  $m_j$  is fitted to the instances within each partition  $p_i$  (i.e.  $D_{p_i}$ ). This forms a set of models,  $M$ . In some cases each model may be associated with a single model, i.e.  $|P| = |M|$ . In other cases, each model may be fitted to the instances from multiple partitions or multiple models may be fitted to the same partition (e.g. different models for subsets of features), i.e.  $|P| \neq |M|$ . When  $|P| < |M|$ , appropriate methods will be required for reconstructing the raw data from multiple models and combining them to output a single instance. Thus, the reduction of  $D$  outputs the set of partitions and set of models,  $\langle P, M \rangle$ . The  $k$ D-STR algorithm, the main contribution of this chapter, outputs  $\langle P, M \rangle$  for a given dataset  $D$  and is discussed further in Section 4.2.

### 4.1.2 Evaluating Reduced Datasets

Several metrics can be used to evaluate the reduction  $\langle P, M \rangle$  of  $D$ . These can be separated into each of the following three categories.

#### Evaluation of Partitioning

The number of partitions,  $|P|$ , is indicative of both the similarity of instances that are near to each other in the spatio-temporal domain, as well as total dissimilarity of all instances in the dataset. Furthermore, the distribution of instances within a partition is indicative of the dissimilarity of instances. For example, partitions that contain many instances spread across a larger spatial area or time period indicate high spatial autocorrelation. Large partitions that contain many similar instances can be accurately modelled using a simple model, resulting in a significant reduction in the storage needed to represent the data. Conversely, small partitions that contain few instances are indicative of low autocorrelation. The number of partitions and their distribution in the spatial and temporal domains is indicative of the autocorrelation in the dataset and amount of storage required to store the dataset. These evaluation criteria may be formalised as:

- (a) the number of partitions, denoted  $|P|$ ,
- (b) the distribution of the number of instances within each partition, denoted  $\{|D_{p_i}| \mid p_i \in P\}$ ,
- (c) the distribution of spatial area covered by each partition and the number of sensors within each partition, denoted  $\{\text{area}(p_i) \mid p_i \in P\}$  and  $\{\text{sensors}(p_i) \mid p_i \in P\}$  respectively, and
- (d) the distribution of the period of time covered by each partition and the number of time intervals within each partition, denoted  $\{\text{time}(p_i) \mid p_i \in P\}$  and  $\{\text{periods}(p_i) \mid p_i \in P\}$  respectively.

#### Evaluation of Modelling

The accuracy of the models stored affects both the reconstruction of the original instances and imputation of instances not present in the input dataset. However, depending on the modelling process used, analysis may be performed using the reduced data without reconstructing the original instances. Therefore, criteria for evaluating the models output are:

- (a) **Reconstruction and Imputation Error**

To measure the accuracy of a partition's model, we can reconstruct each of the instances within the partition and measure the error between

the raw input instance and the reconstructed instances. By measuring the accuracy of all of the models, we can evaluate the accuracy of the entire reduction. One measure of error, the mean average percentage error (MAPE), indicates how incorrect the reconstructed instance is as a percentage of the original value. For a given feature  $f \in F$ , the MAPE metric is defined as:

$$e_{\text{MAPE}}(D, \hat{D}, f) = \frac{1}{|D|} \sum_{d_{s,t} \in D} \left| \frac{d_{s,t}^f - \hat{d}_{s,t}^f}{d_{s,t}^f} \right| \quad (4.2)$$

where  $\hat{D}$  is the dataset of instances reconstructed from  $\langle P, M \rangle$ . Furthermore,  $d_{s,t}^f$  is the value of feature  $f$  for instance  $d_{s,t}$ , and  $\hat{d}_{s,t}^f$  is the value of feature  $f$  for the reconstructed instance  $\hat{d}_{s,t}$ .

However, the MAPE metric is undefined when the value of  $d_{s,t}^f$  is 0. In cases where at least one value is non-zero, the normalised root mean square error (NRMSE) can be used:

$$e_{\text{NRMSE}}(D, \hat{D}, f) = \frac{\psi(D, \hat{D}, f)}{\text{range}(f)} \quad (4.3)$$

$$\psi(D, \hat{D}, f) = \sqrt{\frac{\sum_{d_{s,t} \in D} (d_{s,t}^f - \hat{d}_{s,t}^f)^2}{|D|}}$$

where  $\text{range}(f) = \arg \max_{s,t} (d_{s,t}^f) - \arg \min_{s,t} (d_{s,t}^f)$ .

To evaluate the reconstruction over multiple features, the average error over all of the features can be used. By normalising the error for each feature by the range of values present, the error of two features with different scales can be compared. To calculate the expected error of imputation, one or more instances can be withheld from the input dataset and estimated by inputting its time and location into the model of the applicable partition. By comparing the withheld instance against its reconstructed value we can estimate the imputation error of the reduction.

#### (b) **Explained Variation**

As well as reconstruction error, it is useful to consider how well each model accounts for the variation within its partition(s). One such measure is the explained variance, or the coefficient of determination ( $r^2$ ), which measures the percentage of variance of a feature  $f \in F$  that has been

captured by the model:

$$e_{r^2} = 1 - \frac{\sum_{d_{s,t} \in D} (d_{s,t}^f - \hat{d}_{s,t}^f)^2}{\sum_{d_{s,t} \in D} (d_{s,t}^f - \bar{d}^f)^2} \quad (4.4)$$

By calculating the  $r^2$  value for each partition, we may identify which spatial areas or temporal periods do not accurately account for the variance in  $f$ . However, it is important to consider how the variation of the feature over space and time has been captured. For this purpose we can compare the spatial and temporal variograms for the reconstructed data against the variogram of the original data.

(c) **Descriptive Statistics**

For analysis using descriptive statistics, measuring the difference in descriptive statistics for the entire dataset or particular regions and time periods is useful. Thus, we can measure the change in the statistics presented in Section 3.1.2 to estimate the utility, or expected error, when using the reduced data for analysis.

(d) **Complexity and Diversity of Models**

Combined with reconstruction error, understanding the complexity of each model describes both the variability of the instances within each partition, and how efficient the model is. For example, more complex models in some partitions, relative to the number of instances and the reconstruction error of the instances, may be indicative of high variability in those areas and time periods. The number of coefficients used to store a model  $m_j$ , denoted  $|m_j|$ , measures how complex the model is.

The diversity of models across partitions is also indicative of the variability of the data across space and time, and of patterns that may be present in the data. For example, if partitions with similar models exist at regular intervals, this indicates a repeating pattern in the data. We may then be able to replace those models with a reference to a single model which uses engineered features rather than location in space and time as the independent variables.

One method for comparing model diversity is to use the model of one partition to estimate the values of instances in another partition. If the error incurred is low, the two models can be said to be similar. Other methods for comparing models are dependent on the modelling technique used. That is, in the case of linear regression models we can compare the intercepts and slope coefficients to compare two models. However, in the case of discrete cosine transform (DCT) modelling, we must compare the coefficients of cosine frequencies.



## Evaluation of Storage Used

Each instance in the input dataset, that is the unreduced raw data, requires  $k$  values to define its location in space and time, and  $|F|$  values for its feature responses. Thus, the storage requirement of the input dataset is:

$$\text{storage}(D) = |D| \cdot (k + |F|) \quad (4.5)$$

To store the reduced dataset, the coordinates of each partition's bounds in space and time have to be stored, as well as the coefficients for each model. The temporal bounds of each partition are defined by 2 timestamps, and the spatial bounds are defined by  $|b_i|$  coordinates, each of which is defined by a value in each of the  $\mathcal{D} = k - 1$  dimensions. Thus, the storage requirement of the reduced dataset is:

$$\text{storage}(\langle P, M \rangle) = \sum_{i=1}^{|P|} (|b_i| \cdot (k - 1) + 2) + \sum_{j=1}^{|M|} |m_j| \quad (4.6)$$

where  $|m_j|$  denotes the number of coefficients used to store model  $m_j$ . Note that when a model is stored per feature,  $m_j$  refers to the set of feature models covering the same locations and timestamps. In this case,  $|m_j|$  is the sum of coefficients of those models.

To measure the quantity of storage saved by reducing  $D$  to  $\langle P, M \rangle$ , the ratio between the two metrics can be used. This is denoted:

$$q(D, \langle P, M \rangle) = \frac{\text{storage}(\langle P, M \rangle)}{\text{storage}(D)} \quad (4.7)$$

## 4.2 $k$ D-STR: $k$ -Dimensional Spatio-Temporal Reduction Algorithm

The aim of spatio-temporal dataset reduction is to reduce an input dataset  $D$  to a set of partitions  $P$  and models  $M$ . To accomplish this, the  $k$ D-STR algorithm is proposed, which aims to minimise both the quantity of data used by the reduced dataset compared to the input dataset, and the information lost during the reduction process. In this thesis, the mean NRMSE (Equation 4.3) calculated over all of the features is used as a measure of information lost. The NRMSE metric provides a measure of model accuracy in a way that is agnostic to the later analysis performed using the reduced dataset and, by averaging the NRMSE over all of the features, measures the accuracy of all features relative to the range of their original values. To minimise both the storage used and the information lost by the reduction,  $k$ D-STR minimises the

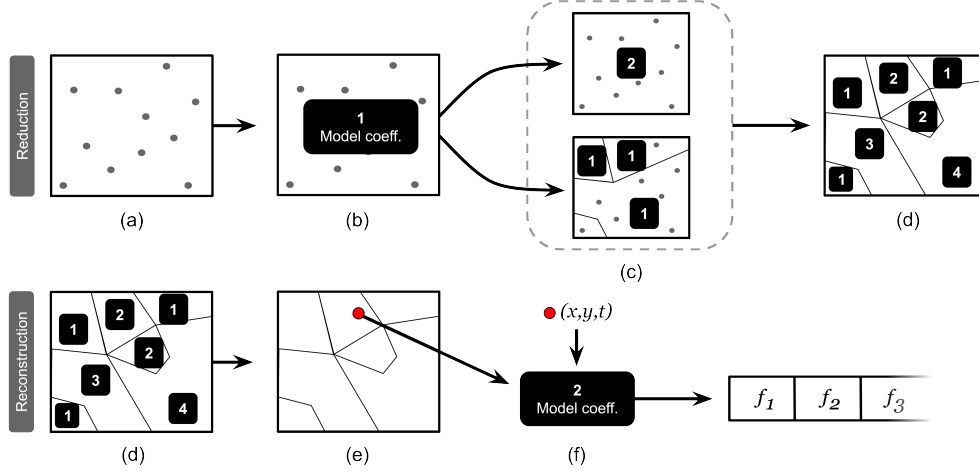


Figure 4.2: Overview of  $k$ D-STR data reduction and reconstruction. (a) The input dataset. (b) Spatial and temporal domains partitioned into 1 partition, and 1 model coefficient stored. (c) Number of model coefficients and number of partitions repeatedly increased and tested until the objective function is minimised. (d) Partitions and model coefficients output. (e) Reconstructing the feature values at a location and time — the containing partition is selected and its model retrieved. (f) Location and time input into model and feature values reconstructed.

*Reconstruction Error and Storage (RES) heuristic:*

$$h(D, \langle P, M \rangle) = \alpha \cdot q(D, \langle P, M \rangle) + (1 - \alpha) \cdot e(D, \langle P, M \rangle) \quad (4.8)$$

where  $e(D, \langle P, M \rangle)$  is a measure of the reconstruction error (i.e.  $e(D, \langle P, M \rangle) = e_{\text{NRMSE}}(D, \langle P, M \rangle)$ ). In this heuristic,  $\alpha$  is a parameter which defines the user's preference for reduction in storage volume versus information lost. The parameter is bound to the range  $(0, 1)$ , and must be determined before reducing the dataset.

The  $k$ D-STR algorithm therefore requires three inputs: (i) the input dataset, (ii) a value for the parameter  $\alpha$ , and (iii) a choice of modelling technique used to model the instances within each partition.  $k$ D-STR is an iterative algorithm that begins by forming a partitioning tree over the dataset. Then, starting with a single partition at the root of the tree, a model is fitted to the instances within the partition to summarise the data. Next,  $k$ D-STR iteratively determines whether to partition the  $\mathbb{S} \times \mathbb{T}$  space into more partitions, or to increase the complexity of one of the existing models with the aim of improving its accuracy. The decision taken at each step is that which minimises the RES heuristic  $h(D, \langle P, M \rangle)$ , and the algorithm terminates when the heuristic cannot be minimised further. An overview of this process is shown in Figure 4.2. When  $k$ D-STR terminates, the algorithm outputs the set of partitions and models, as shown in Figure 4.2(d).

Table 4.1: Example footfall data recorded at the 9 sensors (A-K) shown in Figure 4.1.

		Sensor										
		A	B	C	D	E	F	G	H	I	J	K
<b>Time step</b>	<b>0</b>	252	278	148	193	279	248	267	296	45	241	58
	<b>1</b>	247	305	153	145	301	212	207	292	67	201	52
	<b>2</b>	210	296	139	134	299	199	192	287	39	189	46

In Section 4.2.1, the process of partitioning the spatio-temporal domain of the dataset is presented and, in Section 4.2.2, the process of modelling each partition is discussed. In Section 4.2.3, the iterative steps of the  $k$ D-STR algorithm are given, describing the complete reduction process from input to output. Finally, in Section 4.2.4, the process of reconstructing the data from the reduced form is discussed.

#### 4.2.1 Partitioning Technique of $k$ D-STR

The novel partitioning technique used by  $k$ D-STR uses the difference between instances in the feature space to partition the instances in the spatio-temporal space. The technique gives a hierarchical partitioning of the instances, with a complete dissection of the  $\mathbb{S} \times \mathbb{T}$  space into non-overlapping partitions for each level of the tree. Each partition is therefore defined by a piece-wise linear polygon in the spatial domain and a beginning and end time in the temporal domain.

To begin partitioning the dataset, the instances are first clustered into a set of clusters  $C$  using hierarchical agglomerative clustering in the feature space. By clustering the instances in the feature space,  $k$ D-STR clusters together instances that have similar feature values regardless of when and where the instances were recorded. Hierarchical clustering is used as the resultant tree allows partitions and models to be retained in some regions and time periods as the number of clusters required changes. Furthermore, there is no need to translate the desired number of clusters into a distance value for each level of the partition tree as is required by density based approaches. To illustrate this further, consider the simple example dataset shown in Table 4.1, which contains data from footfall sensors that record the number of people that walk through an area at three consecutive time steps. The data has been hierarchically clustered using only the raw footfall count values. This results in the cluster tree shown in Figure 4.3(a), where each node in the tree shows the boundaries of the clusters in the footfall feature space.

After the cluster tree is formed, partitions are found for each level. For a given level of the cluster tree, each instance in the  $\mathbb{S} \times \mathbb{T}$  space is labelled with

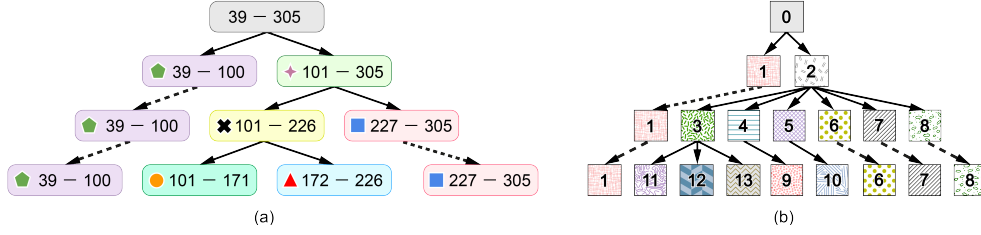


Figure 4.3: Relationship between clusters and partitions in  $k$ D-STR. (a) The cluster tree created by hierarchical clustering on the data in Table 4.1. Each node in the cluster tree shows the bounds defining the cluster. Solid arrows are used to show which clusters decompose into new clusters at each level of the tree, and dashed arrows are used to show clusters that remain the same. (b) The resulting partitioning tree.

the cluster it has been grouped into. Then, homogeneous partitions that are connected components belonging to the same cluster in the  $\mathbb{S} \times \mathbb{T}$  space are found. Since storing the bounding coordinates of each partition may require many values to be stored, we assert that each partition must be defined by a single start and end time to limit the shape that partitions may take in the  $\mathbb{S} \times \mathbb{T}$  space. To create partitions from the clustered instances,  $k$ D-STR first partitions the spatial domain  $\mathbb{S}$  into discrete polygons around each sensor using Voronoi partitioning [14] as shown in Figure 4.1(a). A similar action is also performed in the temporal domain  $\mathbb{T}$ , forming a discrete timestep around each unique timestamp present in  $D$ . By discretising the spatio-temporal domains, the instances in  $D$  can be viewed as a spatio-temporal graph, where each vertex is an instance and edges link vertices that are *adjacent*. Two instances are said to be adjacent if:

- (i) they were recorded consecutively at the same sensor, or
- (ii) they were recorded at the same time and the polygons surrounding their sensors are adjacent in the discretised spatial domain.

After discretising the spatial and temporal domains,  $k$ D-STR extends partitions in a breadth-first manner. First, an instance is chosen as the start of a new partition, and all instances that are adjacent to this instance in the spatial domain and belong to the same cluster are added to the partition. Then, after all spatial neighbours of the initial instance are considered, the temporal boundary is extended by up to 1 timestep before and after the initial instance if doing so does not break the cluster homogeneity of the partition. This process is repeated continuously, expanding the boundaries of the partition by a depth of 1 neighbour to the existing instances in the partition spatially, and 1 timestep before and after the partition, until the spatial and temporal bounds of the partition cannot be expanded further.

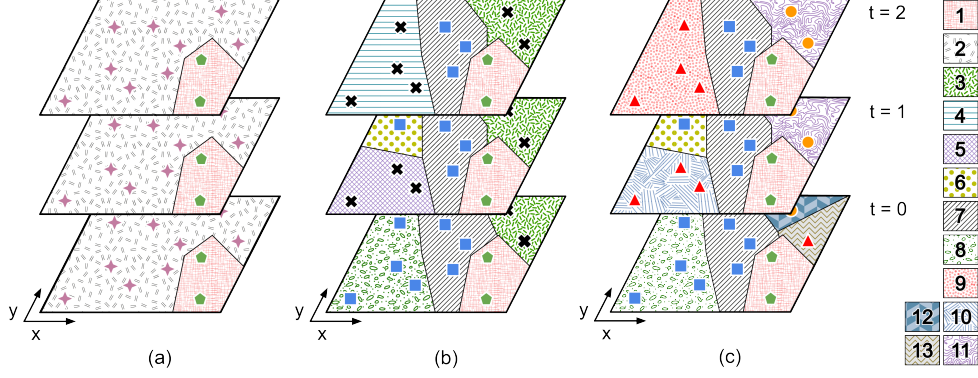


Figure 4.4: The spatio-temporal partitions formed by the data partitioning process, when applied to the data shown in Table 4.1. Subfigures (a), (b) and (c) show the partitions at levels 2, 3 and 4 of the partitioning tree respectively. Subfigure (c) is also shown in Figure 4.1(c), and partitions 1—9 in Figure 4.1(c) correspond to partitions 1 and 6—13 here.

Converting a level of the cluster tree into a level of the partition tree is complete when all instances in  $D$  are associated with a homogeneous partition within that level of the partition tree. The result is a partition tree in which each instance is assigned to a partition at each level of the tree. The relationship between clusters and partitions is shown in Figure 4.3. The first four levels of the cluster tree are shown in subfigure (a), and the corresponding levels of the partitioning tree are shown in subfigure (b). Each level of the partitioning tree is also shown in Figure 4.4. At the root of the cluster tree, at level 1 with 1 cluster, all 33 instances shown in Table 4.1 are placed into a single partition, namely partition 0. On level 2 of the cluster tree, partition 0 is decomposed into partitions 1 and 2 in the partitioning tree. On level 3, the 101—305 cluster has been decomposed into 2 more clusters, and partition 2 has been decomposed into partitions 3—8 respectively. Finally, on level 4 of the cluster tree, the 101—226 cluster has been decomposed and partition 3 decomposed into partitions 11—13, partition 4 into partition 9, and partition 5 into partition 10. While the sensors and timesteps belonging to partitions 4 and 5 do not change, the parent cluster has been decomposed and so these partitions were replaced by partitions 9 and 10 respectively.

By using hierarchical clustering,  $kD$ -STR ensures that only some partitions in the partitioning tree are decomposed between levels. This results in some partitions remaining unchanged throughout multiple levels of the partitioning tree, and this is exploited in the reduction process by retaining models of these partitions during reduction. Furthermore, the models of partitions that are replaced but retain the same sensors and timesteps are also retained.

### 4.2.2 Modelling Techniques for $k$ D-STR

After partitioning  $D$  into a hierarchy of partitions, a technique is required to model the instances within each partition. A model may be formed of the instances within each partition or, since each partition is associated with a cluster, a model may be formed per cluster and all partitions belonging to the same cluster linked to the same model. By storing a model per partition, the number of input instances per model may be lower compared to storing a model per cluster. In general, this results in simpler models and fewer coefficients stored per model. However, when storing a model per cluster, the number of coefficients stored may be fewer than the total number of coefficients stored over all partitions. In both cases, the spatio-temporal boundaries of each partition are stored alongside the coefficients of each model, and each model is associated with either one partition or all partitions that originated from the same cluster. To maximise the utility of the reduction, we require the ability to reconstruct the instances of the input dataset from the models output by  $k$ D-STR. Furthermore, we wish to enable the imputing of instances at spatial and temporal locations that have not been sampled in the input dataset, but have nearby sensors and timesteps.

For each partition  $p_i$  or cluster  $c_i$  in each level of the partitioning tree, a model  $m_j$  is fitted to the instances within the partition or cluster. The spatial and temporal values of the instances are used as the independent or predictor values of the model, while the feature values of the instances are used as the dependent or response values of the model, that is,  $m_j : \mathbb{R}^k \rightarrow \mathbb{R}^{|F|}$ . For example, in the case of polynomial regression (PLR) when  $k = 3$ , the model  $m_j$  may be:

$$d_{s,t}^f = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 t + \epsilon \quad (4.9)$$

where  $x$  and  $y$  are the spatial coordinates of  $d_{s,t}$ , i.e.  $(x, y) = s$ . Here, the feature values in the dataset  $D$  are modelled as linear outputs of the spatial and temporal positions of each instance. Thus, the modelling step computes and stores the model coefficients  $\beta_0, \beta_1, \beta_2$  and  $\beta_3$ .

However, in many cases, the feature values may not be linearly dependent on the spatial and temporal positions of the instances, and so adding interaction terms to the model may improve its accuracy. For PLR, this is achieved by increasing the *degree* of the model, although this increases the number of model coefficients stored. Thus, on each iteration,  $k$ D-STR tests the change in  $h(D, \langle P, M \rangle)$  (Equation 4.8) when the degree of the model is increased by 1.

This thesis considers three illustrative modelling techniques. These techniques were chosen for their ease of understanding as well as their differing approaches for modelling spatio-temporal data. First, multivariate polynomial

regression (PLR) is considered owing to its ability to explain data that is spatially and temporally autocorrelated. The model equation used for 1 degree of freedom is shown in Equation 4.9. More information about PLR can be found in literature [114]. Second, multidimensional discrete cosine transform (DCT) approximation is considered for its ability to model periodic data using few coefficients. By removing low-weighted coefficients and storing just those that are weighted highly, the original data can be reproduced with reasonable accuracy. A detailed description of multidimensional DCT approximation and data reconstruction can be found in [103]. Finally, decision tree regression (DTR) is considered owing to the interpretability of the models output. The DTR method forms a decision tree over the spatial and temporal features and, for each leaf node in the tree, computes a regression tree for the instances belonging to that node. Furthermore, the method uses the *maximum depth* of the tree as a method for controlling the number of regression models computed and regression coefficients stored. Further information on DTR modelling can be found in [24].

For each of these techniques, we may form a model per partition or a model per cluster. Thus, the techniques considered in this thesis are referred to as PLR-P and PLR-C, DCT-P and DCT-C, and DTR-P and DTR-C respectively. Other modelling techniques may be applicable and can be considered, and the techniques described here are considered for illustrative purposes. If the type of analysis to be performed after reduction is already known, more appropriate modelling techniques may be used that offer higher accuracy, smaller storage or permit faster answering of queries. Overall, data reduction aims to use less storage for each model than the input dataset uses to store the instances the model is fitted to. Furthermore, the aim of  $k$ D-STR is to use the independent and dependent variables described above for the modelling process.

### 4.2.3 Data Reduction Algorithm

The  $k$ D-STR algorithm can be seen in Algorithm 1, in which a model is formed for each partition, although the algorithm may be easily adapted to model on clusters instead. Whether forming a model per partition or per cluster,  $k$ D-STR outputs both the spatial and temporal boundaries of each partition and the coefficients of each model. In the case of modelling on partitions, a model is fitted to the instances within each partition and each model is associated with a single partition. In the case of modelling on clusters, a model is fitted to the instances of all partitions associated with each cluster and each model is associated with one or more partitions.

Prior to initialising  $k$ D-STR, a value for the parameter  $\alpha$  must be chosen, where  $0 < \alpha < 1$ . This parameter weights the error introduced against the

storage ratio in the RES heuristic (Equation 4.8). Since a value of 0 would indicate a preference for minimising the error introduced with no consideration of the storage used,  $k$ D-STR would not stop iterating until the error metric  $e(D, \langle P, M \rangle) = 0$ . Such a perfect model may be unrealistic and less efficient than storing the original input dataset. Similarly, if  $\alpha = 1$ , both increasing the complexity of a model and increasing the number of partitions will increase the storage used beyond that of the initialisation step. Therefore,  $k$ D-STR would not iterate beyond the first iteration. Both of these scenarios may not be useful and so values of 0 and 1 are not considered for the parameter  $\alpha$ .

After a partitioning tree has been formed for  $D$  (Algorithm 1, lines 1–3), and a value of  $\alpha$  and a modelling technique has been chosen, the reduction steps of  $k$ D-STR are performed.  $k$ D-STR begins at the root of the partitioning tree, with all instances belonging to a single partition. This partition is then modelled using the simplest form of the modelling technique chosen (lines 5–7): in the case of polynomial regression (PLR), a polynomial model of order 0 (simply a mean value) is constructed for each feature; in the case of DCT, only the highest weighted cosine coefficient is considered; in the case of DTR, the decision tree is limited to a depth of 1. Note that the value of 1 is used in line 6 to indicate the simplest form of model. After the model is fitted to the data, the result of the RES heuristic  $h(D, \langle P, M \rangle)$  is calculated for the first reduction step (line 8).

After the first reduction step, the algorithm iterates. On each iteration,  $k$ D-STR decides whether to increase the complexity of one of the existing models (lines 11–18), or increase the number of partitions in the partitioning of the  $\mathbb{S} \times \mathbb{T}$  space (lines 19–27). When the number of partitions is increased, only some existing partitions are decomposed. For partitions that are not decomposed, the models for these partitions persist, thereby improving the efficiency of  $k$ D-STR (lines 22–23). For partitions that are decomposed, new partitions are found and new models are fitted to these subsets of the data (lines 24–26).

The  $k$ D-STR algorithm terminates when the RES heuristic cannot be minimised further, i.e.  $h_1 \geq h(D, \langle P, M \rangle)$  and  $h_2 \geq h(D, \langle P, M \rangle)$ . When the algorithm terminates the reduction is complete and the set of partitions and set of models,  $\langle P, M \rangle$ , is output.

#### 4.2.4 Data Reconstruction

To reconstruct an instance from the reduced dataset  $\langle P, M \rangle$ , the location and time of the instance is required,  $(s, t)$ . First, the partition containing  $(s, t)$  is selected (as shown in Figure 4.2(e)). Then, the selection partition’s model is retrieved and  $(s, t)$  is input into the model (as shown in Figure 4.2(f)). The



---

**Algorithm 1:**  $k$ D-STR algorithm for reducing spatio-temporal data

---

**Data:** A spatio-temporal dataset  $D$

**Result:** Reduced dataset  $P$ ,  $M$

```
1 clusterTree = cluster( $D$ )
2 numberClusters = 1
3  $P \leftarrow \text{findPartitions}(D, \text{clusterTree}, \text{numberClusters})$ 
4  $M \leftarrow \emptyset$  // Initialise the set of models to the empty set
5 for  $p_i$  in  $P$  do
6    $m_i \leftarrow \text{model}(D_{p_i}, 1)$  // Model partition  $p_i$  using 1 coefficient
7    $M.\text{add}(m_i)$ 
8  $h \leftarrow \text{heuristic}(D, P, M)$  // Heuristic for 1 partition and 1 coeff.
   // Now iterate until heuristic  $h$  is minimised
9 do
   // First, check if increasing an existing model's
   // complexity minimises  $h$  further
10   $h_1 \leftarrow h$ 
11  for  $p_i$  in  $P$  do
12     $M' \leftarrow M$ 
13     $m'_i \leftarrow \text{model}(D_{p_i}, m_i.\text{complexity} + 1)$ 
14     $M'.\text{replace}(m_i, m'_i)$  // Replace  $m_i$  with  $m'_i$  in  $M'$ 
15     $h' \leftarrow \text{heuristic}(D, P, M')$ 
16    if  $h' < h_1$  then
17       $h_1 \leftarrow h'$ 
18       $M_{\text{best}} \leftarrow M'$ 
   // Second, check if increasing number of partitions minimises  $h$ 
19   $P' \leftarrow \text{findPartitions}(D, \text{clusterTree}, \text{numberClusters}+1)$ 
20   $M'' \leftarrow \{\}$  // Initialise the set of models to the empty set
21  for  $p_i$  in  $P'$  do
22    if  $p_i$  in  $P$  then
23       $M''.\text{add}(m_i)$  // Add  $m_i \in M$  to  $M''$ 
24    else
25       $m''_i \leftarrow \text{model}(D_{p_i}, 1)$ 
26       $M''.\text{add}(m''_i)$ 
27   $h_2 \leftarrow \text{heuristic}(D, P', M'')$ 
   // Finally, if increasing the number of partitions or complexity
   // of a model is more optimal, take that choice
28  if  $h_1 < h_2$  and  $h_1 < h$  then
29     $M \leftarrow M_{\text{best}}$ 
30     $h \leftarrow h_1$ 
31  else if  $h_2 < h_1$  and  $h_2 < h$  then
32     $P \leftarrow P'$ 
33     $M \leftarrow M''$ 
34     $h \leftarrow h_2$ 
35    numberClusters  $\leftarrow$  numberClusters + 1
36 while  $h_1 < h$  or  $h_2 < h$ 
37 return  $P$ ,  $M$ 
```

---

output of the model is the reconstructed feature values of  $D$  at  $(s, t)$ . For example, in the case of PLR, the instance can be reconstructed using Equation 4.9. Note that an instance at that location and time may not have been present in the input dataset prior to reduction. This allows the data at a location and time to be imputed directly using the reduced dataset without requiring multiple other instances be reconstructed as is required by IDEALEM and ISABELA.

#### 4.2.5 Analysis of Complexity

To understand the time and memory complexities of  $k$ D-STR, the startup cost of clustering the dataset and the cost of each iteration of the algorithm have to be considered. In the startup phase of the algorithm (line 1 of Algorithm 1), hierarchical clustering is performed. While hierarchical agglomerative clustering runs in  $\mathcal{O}(|D|^3)$  time and requires  $\mathcal{O}(|D|^2)$  memory, a more efficient approximation, which has been shown to yield comparable clusters, has been demonstrated to reduce the time complexity to  $\mathcal{O}(|D|^2)$  [92]. Therefore, the startup phase of  $k$ D-STR requires  $\mathcal{O}(|D|^2)$  time and  $\mathcal{O}(|D|^2)$  space.

After clustering, the algorithm iterates in successive rounds, either increasing the number of partitions in the reduced dataset or increasing the complexity of a model. To increase the number of partitions, the number of clusters is increased by 1. Each instance is labelled with its cluster number in the next level of the cluster tree, and homogeneous partitions in the  $\mathbb{S} \times \mathbb{T}$  space are found for each cluster at this next level. To create a partition, an instance that has not been assigned to a partition is chosen at random as the start of the next partition, and other instances adjacent to the instance are added to the partition if they belong to the same cluster. This process is repeated in rounds until no more instances can be added to the partition. During this process, the boundary between any two adjacent instances is checked at most twice. First, the boundary is checked in one direction to see if the two instances belong to the same cluster. If they do not, the second instance is not added to the partition. In this case, at a later time of processing the second instance will be added to a different partition and the same boundary is checked again to see if the first instance can be added to the new partition. Since that instance already belongs to a partition it will not be added, but the adjacency edge between the two instances has now been considered twice. This process requires  $\mathcal{O}(x|D|)$  time, where  $x$  is the maximum number of instances that are adjacent to any single instance in  $D$ . Furthermore, this process requires  $\mathcal{O}(|D|)$  memory to store the cluster and partition number of each instance, as well as a working list of the boundary instances of the partition being expanded.

As well as increasing the number of partitions on each iteration, the

algorithm also considers increasing the complexity of an existing model. For example, in the case of polynomial linear regression (PLR), this requires  $\mathcal{O}(y^2|D|)$  time and  $\mathcal{O}(y^2)$  memory per model, where  $y$  is the number of coefficients calculated for the model. Discrete cosine transforms (DCT) can be performed in  $\mathcal{O}(|D|^2)$  time and  $\mathcal{O}(1)$  memory per model, although fast cosine transforms can be performed in  $\mathcal{O}(|D| \log |D|)$  time. Similarly, decision tree regression (DTR) can be performed in  $\mathcal{O}(k|D|^2)$  time and  $\mathcal{O}(k)$  memory per model. Therefore, the startup phase of  $k$ D-STR requires  $\mathcal{O}(|D|^2)$  time and  $\mathcal{O}(|D|^2)$  memory, while each iteration, in the case of PLR, requires  $\mathcal{O}((x + y^2|M|)|D|)$  time and  $\mathcal{O}(|D| + y^2|M|)$  memory.

### 4.3 Experimental Evaluation Methodology

To evaluate  $k$ D-STR, the datasets presented in Chapter 3 are reduced using different values of the parameter  $\alpha$ . The values of  $\alpha$  tested are  $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ , providing a range of values within the bounds  $(0, 1)$ . For each of the datasets, a range of values of the parameter  $\alpha$  are tested to evaluate hypotheses on the relationship between  $\alpha$  and the reduced output. Furthermore, each of the 3 modelling techniques presented in Section 4.2.2 are evaluated, as well as modelling on both each partition and each cluster. This results in 6 modelling methods:

- (a) Polynomial linear regression on each partition (PLR-P),
- (b) Polynomial linear regression on each cluster (PLR-C),
- (c) Discrete cosine modelling on each partition (DCT-P),
- (d) Discrete cosine modelling on each cluster (DCT-C),
- (e) Decision tree regression on each partition (DTR-P),
- (f) Decision tree regression on each cluster (DTR-C).

The metrics used to evaluate the reduction are those presented in Section 4.1.2. That is, the ratio of storage used and NRMSE relative to the input datasets are used to measure the efficiency and error of the reduced datasets. Furthermore, the number of partitions in space and time are compared with the variance of the datasets in space of time. Together, these metrics evaluate the partitioning used in the reduced dataset, the storage used compared to the input dataset, and the error incurred in the reconstructed data.

Several hypotheses are tested, namely:

*H1.1 Low values of  $\alpha$  result in low NRMSE but high storage used, and vice versa; data samples with high spatio-temporal variance require more*

*storage or incur a higher NRMSE.*

For values of  $\alpha$  close to 0, the RES heuristic (Equation 4.8) minimises the NRMSE incurred while allowing a high quantity of storage to be used for the reduced dataset. Conversely, for values of  $\alpha$  close to 1,  $k$ D-STR minimises the storage used while allowing the NRMSE to increase. To achieve approximately the same NRMSE, samples of data that have a higher variance require a higher storage volume than data samples with lower variance. Conversely, to achieve approximately the same storage ratio, data samples with a higher variance result in higher NRMSE. Data samples that have a higher variance in space than time result in more partitions in the spatial domain than the temporal domain, and vice versa. For high variance samples, the data may be captured by increasing the number of partitions or increasing the number of model coefficients stored.

- H1.2 *For values of  $\alpha$  close to 0, modelling on clusters may be more efficient than modelling on partitions, although both yield similar results for values of  $\alpha$  close to 1.*

When modelling on clusters, a model is formed for larger subsets of instances than modelling on partitions. When the variability of the data is high, this requires more model coefficients to capture the variability. However, when the variability is low (for example, a cluster of just the 0 mm instances in the rainfall dataset), few coefficients may accurately capture the variability. Therefore, modelling on clusters requires fewer model coefficients than modelling on partitions. This intuition is only applicable when the number of partitions stored (and clusters stored) is greater than 1. When only 1 partition is stored, modelling on either clusters or partitions yields the same result.

- H1.3 *1-Dimensional SRS requires less storage than a 2-Dimensional SRS, although the number of partitions output may be higher.*

The number of coordinate values stored for the bounding polygon  $b_i$  of a polygon  $p_i$  is directly proportional to the number of spatial dimensions stored. Therefore, using a 1-Dimensional SRS requires fewer coordinate values to be stored. This hypothesis applies directly to the traffic datasets which can be represented using a 1D or 2D SRS. However, when sensors are ordered by their distance from a fixed origin, slip roads, which exhibit different traffic characteristics than the main carriageway, act as discontinuities along the single dimension. Thus, a single partition on the main carriageway that is adjacent to a slip road partition in a 2-dimensional SRS will be split into 2 separate partitions in the 1-dimensional SRS. Therefore, the number of partitions may be higher

when a 1-dimensional SRS is used compared to a 2-dimensional SRS.

H1.4 *As the quantity of input data increases, the reduced dataset becomes more efficient with respect to storage for datasets with low variation, and remains approximately the same for datasets with high variation.*

Datasets with low variability yield a sublinear increase in the number of partitions stored as the number of instances increases. Furthermore, they can be accurately modelled using few model coefficients. Thus, increasing the number of time intervals or sensors in the dataset leads to a sublinear growth in the quantity of data stored for the reduced dataset. This assumes the density of instances remains approximately the same in space and time, and the variability of the data does not increase. For high variability datasets, the number of partitions continues to grow sublinearly, however the partitions may have boundary shapes that require many coordinates to be stored. Furthermore, the number of model coefficients required may increase as well. Therefore, the storage ratio of the data may not decrease as the number of instances increases.

As well as assessing the storage used and NRMSE incurred, the impact of dataset size is examined. In the temporal domain, the dataset sizes are made larger by increasing the number of days in the dataset, thereby maintaining the spatio-temporal density while increasing the data size. The sizes tested are  $0.25\times$ ,  $0.5\times$ ,  $1.0\times$ ,  $1.5\times$  and  $2.0\times$ , corresponding to approximately 7.5, 15, 30, 45, and 60 days of data. In the spatial domain, only smaller samples are evaluated ( $0.25\times$ ,  $0.5\times$ ,  $1.0\times$ ) as no further sensors existed.

Finally, to compare  $k$ D-STR with other reduction methods for spatio-temporal datasets, three methods are considered. First, IDEALEM is used as the state-of-the-art in spatio-temporal data reduction [79, 140]. Second, PCA adapted for spatio-temporal sensor data is used owing to its ubiquity as a data reduction method [43]. PCA maps the original instances into a difference feature space, allowing the data to be stored in a smaller form by storing a smaller number of features than the features in the input dataset. Finally, the DEFLATE compression algorithm is used as a benchmark to show the reduction in storage achievable while losing no information and incurring no error [45]. While DEFLATE is able to compress the data to a form smaller than the input dataset, the entire dataset must be decompressed before the data can be used for analysis. Thus, the user must have sufficient memory to store the decompressed raw data and cannot use the compressed dataset directly. Therefore, DEFLATE is included as an indicator of the potential reduction that is achievable using lossless compression algorithms, rather than as a directly comparable reduction method.

## 4.4 Results

In this section, the results of applying  $k$ D-STR to the datasets discussed in Chapter 3 are discussed. A comparison is given of the results for different modelling techniques and the effects of the parameter  $\alpha$ . Furthermore, a discussion is presented on the choice of spatial referencing system (SRS) and the effect of increasing the quantity of input data, as well as the spatial and temporal properties that are found using the partitioning of space and time by  $k$ D-STR.

### 4.4.1 Trade-Off Between Error and Storage

For each of the datasets used, the mean and range of the NRMSE and storage ratios of the  $k$ D-STR reduction are shown in Figure 4.5. A subfigure is shown per dataset, with each showing the results of using each of the 6 modelling techniques discussed in Section 4.3. Furthermore, each subfigure shows the results of the reduction process given the 5 values of the parameter  $\alpha$ .

Several conclusions can be drawn from these results. In all cases, the error introduced by the reduction is smaller for values of  $\alpha$  close to 0 than values close to 1. Conversely, the storage ratio of the data output is greater for values of  $\alpha$  close to 0 than values close to 1. For example, for PLR-P modelling on the air temperature dataset, the storage quantity used for the reduced datasets ranges from 22.9% to 45.2% when  $\alpha = 0.1$ , but decreases to 0.01% when  $\alpha = 0.9$ . Conversely, the error introduced ranges from 3.4% to 6.4% when  $\alpha = 0.1$ , but increases 9.7% to 14.3% when  $\alpha = 0.9$ . A similar relationship between the error introduced and storage quantity used is observable across all of the datasets. As the value of  $\alpha$  increases, the number of partitions output decreases to 1 and the number of coefficients stored per partition varies accordingly (as discussed in Chapter 4.4.2).

A more detailed examination of how the number of partitions and model coefficients output varies with  $\alpha$  is shown in Appendix A.1. Furthermore, the error incurred by the models when  $\alpha \in \{0.1, 0.9\}$  can be seen in Appendix A.2.

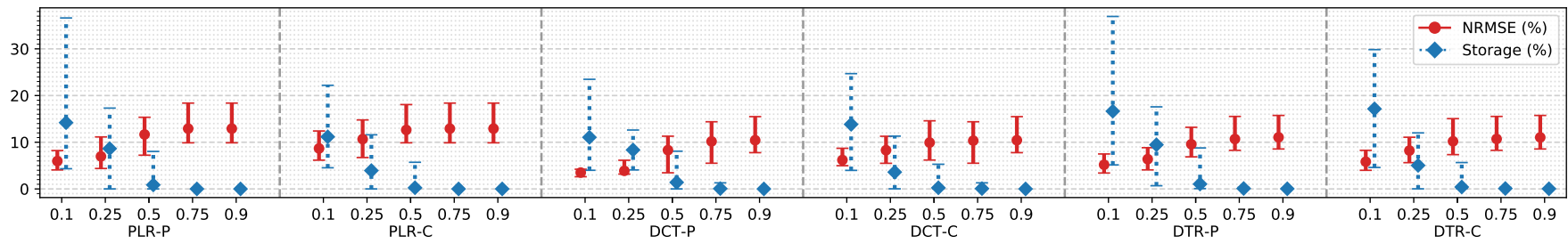
In some cases, particularly when modelling using polynomial regression and when the variability of the data in space and time is low, this trend plateaus after  $\alpha$  increases beyond a certain value. For example, in the case of PLR-P and PLR-C on the air temperature and rainfall datasets, the value of  $\alpha$  has little effect when  $\alpha > 0.25$  and  $\alpha > 0.1$  respectively. This is attributed to the cost of increasing the number of partitions being too great, and the benefits of increasing model complexity of the partitions being insufficient to lower the heuristic value further. Furthermore, as the value of  $\alpha$  increases, increasing the number of partitions to capture the spatio-temporal variability is not worthwhile, and so only a single partition with few model coefficients

is output. The exception to this is DTR modelling on the air temperature dataset. As the value of  $\alpha$  increases, the NRMSE observed also increases and does not appear to plateau for the range of  $\alpha$  values tested.

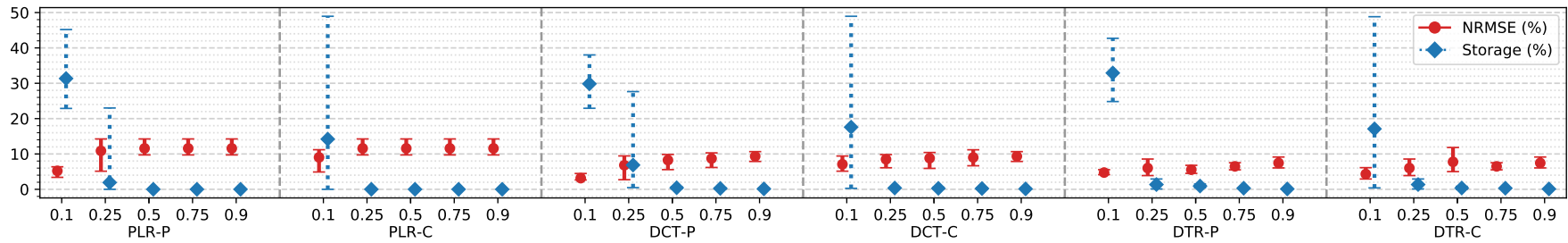
The range of NRMSE and storage ratios also differs between datasets for the same modelling technique and value of  $\alpha$ . When  $\alpha = 0.1$ , the mean storage used is highest for the air temperature dataset, although for  $\alpha > 0.1$  the mean storage used is highest for the traffic datasets. For all values of  $\alpha$ , the mean NRMSE is highest for the traffic datasets and lowest for the rainfall dataset. These results are attributable to the traffic datasets being the most varied in space and time, and the rainfall dataset being the least varied. Furthermore, for the traffic dataset, discontinuities in space result in many partition boundaries in space, often yielding a higher number of partitions relative to the number of sensors present. This is discussed further in Section 4.4.3.

Finally, the relationship between storage and NRMSE differs between the modelling techniques and datasets tested. In most cases, as  $\alpha$  increases, the storage used decreases in an exponential manner. However, the NRMSE appears to follow a log-like pattern in some cases (modelling on clusters for the traffic dataset, PLR and DCT on the air temperature dataset, and all techniques on the rainfall dataset), quadratic pattern in some cases (DTR on the air temperature dataset) and an arctan pattern in others (modelling on partitions for the traffic dataset). This suggests that the value of  $\alpha$  used should be chosen after an initial investigation of the data and, since similar relationships between NRMSE and storage ratio are observed across all samples of the same dataset given a particular modelling technique, the relationship between NRMSE, storage ratio and  $\alpha$  is predictable once a subset of the data has been tested.

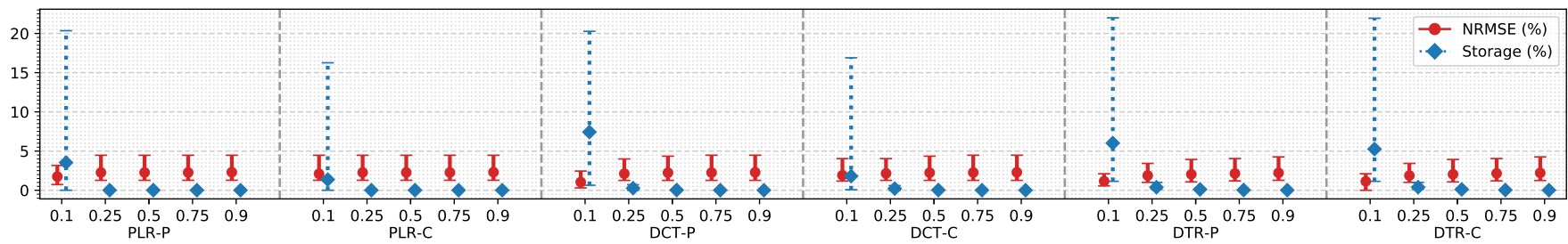
As well as the metric defined in Equation 4.7 for describing the storage used, we may also consider the number of bytes used by the raw and reduced data files. When  $\alpha = 0.1$ , the traffic datasets are reduced to between 679 KB and 69.3 MB, the air temperature datasets are reduced to between 2 KB and 25.8 MB, and the rainfall datasets are reduced to between 2 KB and 11.9 MB. Conversely, when  $\alpha = 0.9$ , the traffic datasets are reduced to between 2 KB and 32.3 MB, the air temperature datasets are reduced to between 2 KB and 5.2 MB, and the rainfall datasets are reduced to between 2 KB and 3.1 MB. However, these reduced data files use the Python pickling process which incurs a significant storage overhead. Consequently, these files may be reduced to a smaller file size by using a less verbose data format, and a direct comparison between the input and reduced file sizes in bytes may not be appropriate.



(a) Traffic



(b) Air Temperature



(c) Rainfall

Figure 4.5: NRMSE incurred and storage used by the reduced traffic, air temperature and rainfall datasets.



#### 4.4.2 Choice of Modelling Technique

From the results presented in Figure 4.5, several conclusions about the choice of modelling technique can be drawn. First, when  $\alpha > 0.25$ , modelling on partitions or clusters has little impact on the storage used and NRMSE achieved. For example, for the traffic datasets, the mean NRMSE when  $\alpha = 0.5$  is 11.7% for PLR-P and 12.6% for PLR-C. Likewise, the mean storage used is 0.9% and 0.3% respectively. However, when  $\alpha \in \{0.1, 0.25\}$ , the NRMSE achieved when modelling on partitions is lower or equal to modelling on clusters for the same modelling technique. For example, when  $\alpha = 0.1$ , PLR-P yields a mean NRMSE of 5.2% and PLR-C yields a mean NRMSE of 9.0% for the air temperature dataset. The only exception to this is when  $\alpha = 0.1$  and DTR modelling is used for the air temperature dataset, with DTR-P yielding a NRMSE of 4.7% and DTR-C yielding 4.3%. Though these lower NRMSE values come at the cost of increased storage, the NRMSE-per-percentage-storage-used, that is the NRMSE divided by storage used, is lower when modelling on partitions versus clusters in almost all cases. Therefore, regardless of the variability of the data in space and time, the benefits of storing a model per partition rather than storing a model per cluster outweigh the increased storage requirements of doing so.

Second, the mean number of partitions output is lower when modelling on clusters versus modelling on partitions, as shown in Table 4.2. For example, for the rainfall dataset when  $\alpha = 0.1$ , PLR-C output 512 partitions while PLR-P output 1215 partitions. The only exception to this is for the traffic datasets when  $\alpha = 0.1$ , where DCT-C modelling yields 8165 partitions and DCT-P modelling yields 6468 partitions. However, modelling on partitions stores a greater or equal number of model coefficients than modelling on clusters in all cases. For most results, the lower NRMSE of modelling on partitions may be attributed to storing more partitions and therefore more model coefficients, thus yielding a more accurate representation of the input data. In such cases, regardless of the variability of the data in space and time, storing more partitions with few model coefficients per partition gives a more accurate reduction of the dataset than storing few models with many coefficients per model. One exception to this conclusion occurs for the air temperature dataset when  $\alpha = 0.1$ . Here, DTR-C modelling yields a marginally lower NRMSE than DTR-P (4.3% versus 4.7%) despite storing 4190 partitions versus 9598 partitions.

Third, as the value of  $\alpha$  increases, the number of coefficients stored per partition decreases, as shown in Figure 4.6. However, as  $\alpha$  increases, the number of partitions output also decreases in some cases, and when fewer partitions are output the number of coefficients stored per partition may

increase. For example, when  $\alpha = 0.1$  and DCT-P modelling is used, 9618 partitions are output on average and the maximum number of coefficients output for a partition is 10. Yet, when  $\alpha = 0.25$ , 2224 partitions are output and the maximum number of coefficients output for a partition is 1330. When  $\alpha > 0.25$ , only 1 partition is output, and the number of coefficients output is 1540, 455 and 286 when  $\alpha = 0.5$ ,  $\alpha = 0.75$  and  $\alpha = 0.9$  respectively. A more detailed examination of how the number of partitions and model coefficients output varies with  $\alpha$  is shown in Appendix A.1.

Fourth, the choice between PLR, DCT and DTR is dependent on the user’s preferences. For example, for all three datasets, using DCT modelling results in a lower NRMSE than PLR, although this comes at the cost of more storage than PLR for the rainfall dataset. However, the difference in NRMSE is often quite small, with the largest difference occurring when  $\alpha = 0.25$  on the air temperature dataset. In this case, the mean NRMSE reported is 4.0% lower for DCT-P compared to PLR-P. Conversely, the mean storage used is 5.0% higher for DCT-P compared to PLR-P. Decision tree regression, which yields easy to interpret models that can be output as a set of if-then rules, achieves similar NRMSE values on all three datasets compared to DCT and PLR. However, this comes at the cost of an increased storage overhead in most cases. Therefore, when a model that is easy to interpret is required, DTR modelling is preferable, however more efficient modelling can be achieved using PLR and DCT.

Table 4.2: Mean number of partitions output by  $k$ D-STR. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

<b>Traffic Dataset</b>						
$\alpha$	PLR-P	PLR-C	DCT-P	DCT-C	DTR-P	DTR-C
0.1	8694	7193	6468	8165	9084	7423
0.25	5182	2359	5076	1749	4971	2263
0.5	407	93	648	22	297	50
0.75	1	1	1	1	1	1
0.9	1	1	1	1	1	1

<b>Air Temperature Dataset</b>						
$\alpha$	PLR-P	PLR-C	DCT-P	DCT-C	DTR-P	DTR-C
0.1	9921	4627	9618	5623	9598	4190
0.25	637	1	2224	1	1	1
0.5	1	1	1	1	1	1
0.75	1	1	1	1	1	1
0.9	1	1	1	1	1	1

<b>Rainfall Dataset</b>						
$\alpha$	PLR-P	PLR-C	DCT-P	DCT-C	DTR-P	DTR-C
0.1	1215	512	2448	513	1377	509
0.25	3	3	3	3	3	3
0.5	3	3	3	3	3	3
0.75	3	3	3	3	3	3
0.9	1	1	1	1	3	1

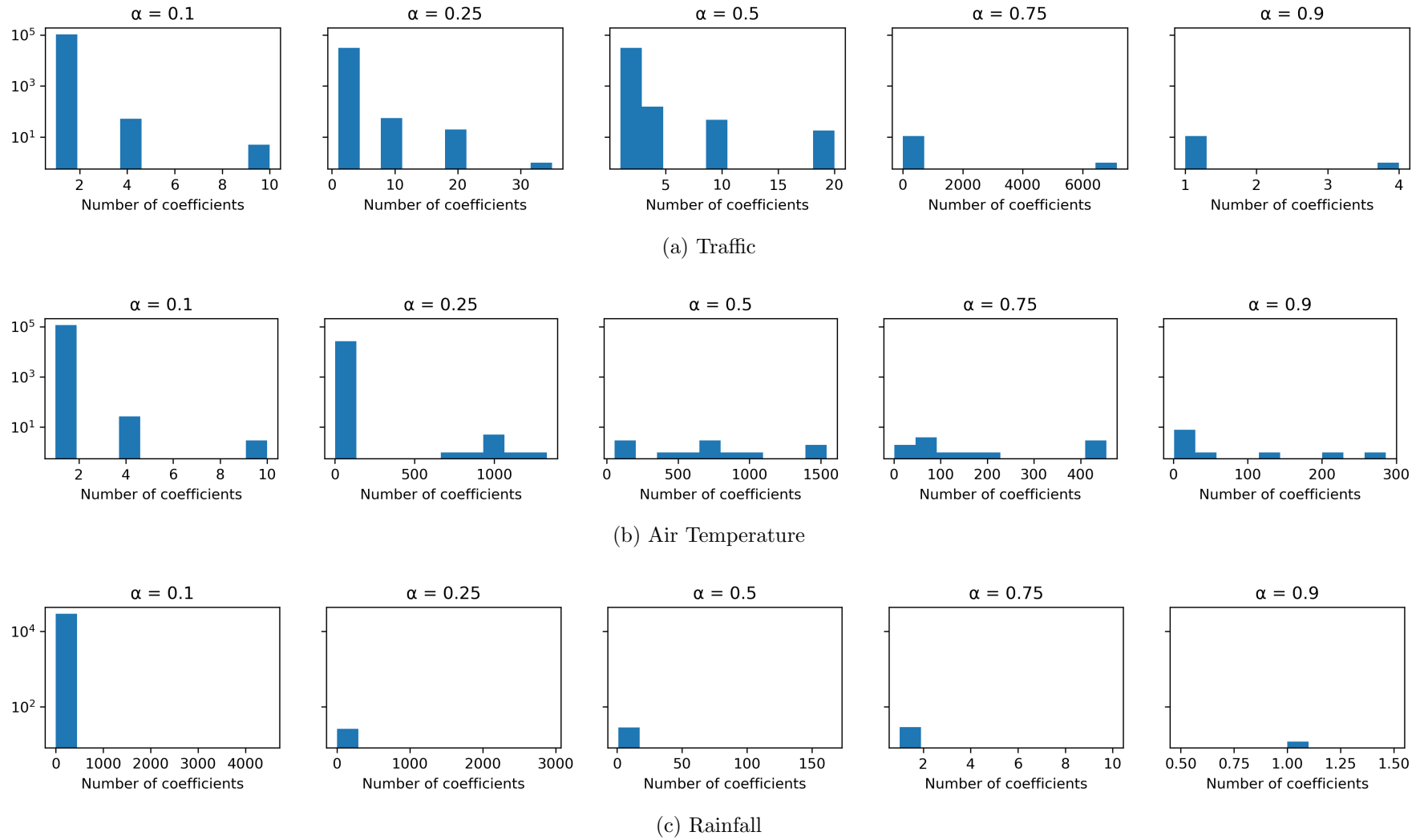


Figure 4.6: Histogram of the number of coefficients output per partition. Results are shown for  $k$ D-STR with DCT-P modelling.

#### 4.4.3 Spatial and Temporal Properties Discovered

The partitioning method used by  $k$ D-STR identifies several patterns and nuances within each of the datasets tested. For example, for the air temperature dataset, several partitions are created that contain either a majority of the sensors for short periods of time, particularly at night, or most of the time intervals in the dataset for a small subset of the sensors. This can be attributed to low spatial variance and low temporal variance at those times and sensors respectively. For the traffic dataset, main carriageway and slip-road sensors are consistently placed in different partitions during the daytime. This distinction is a result of the spatial discontinuity between the main carriageway and slip-roads which results in significant differences in the feature values recorded by the two types of sensor.

For both datasets, the partitioning of the data in time approximately follows the expected daily patterns of the data. That is, partitions of low traffic volume and air temperature are created at night-time, and partitions of high traffic and temperature are created during the day. Similarly, more partitions are created in time as the feature values change in short periods of time at the start and end of the day, and partitions of long periods of time are created during the day. However, this predictability does not hold for the rainfall dataset. The rainfall dataset exhibits discontinuities at times and locations of rainfall versus no rainfall, meaning the locations of partitions in space and time do not follow patterns.

Overall, the number of partitions created in the spatial and temporal domains is approximately related to the variability of the datasets in space and time. For example, when  $\alpha = 0.1$ , the mean time captured per partition is 0.9 hours (approximately 4 time intervals) for the traffic datasets and 2.1 hours (approximately 2 time intervals) for the air temperature dataset (as shown in Appendix A.3). Similarly, the average maximum distance between sensors captured is 1.2 miles and 70 miles. However, for the traffic datasets, many partitions contained many adjacent sensors located on the main carriageway, and many partitions are created for slip roads containing few sensors for long periods of time. Thus, partitions are created in the areas and time periods of high variability, and vice versa. Furthermore, when patterns of high or low feature values exist in the data, these will be discovered by  $k$ D-STR.

Finally, the choice of distance metric used when clustering the data is found to have some impact on the resulting number of partitions output. For example, when  $\alpha = 0.1$  and the M1 traffic dataset is partitioned, the mean number of partitions output is 7,857 for the  $L_1$  (Manhattan) distance metric, 7,353 for the  $L_2$  (Euclidean) distance metric, and 5,661 for the  $L_\infty$  (Chebyshev) distance metric. However, while the number of partitions can vary, the relationship

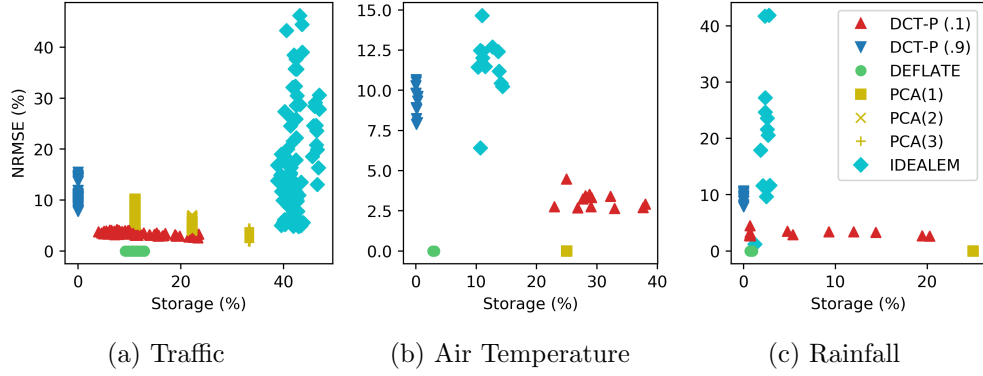


Figure 4.7: Comparison between  $k$ D-STR and PCA, IDEALEM and DEFLATE. Results are shown for  $k$ D-STR using DCT-P modelling with  $\alpha = 0.1$  and  $\alpha = 0.9$ .

between the number of partitions, storage used and error incurred remains the same. This is shown in Appendix A.4.

#### 4.4.4 Comparison with Existing Techniques

A comparison of  $k$ D-STR with existing techniques can be seen in Figure 4.7. In this figure, results for  $k$ D-STR with DCT-P modelling are used for comparison with other techniques<sup>1</sup>. Furthermore, a comparison of the reduced file sizes output by the techniques can be seen in Table 4.3. Compared to  $k$ D-STR, DEFLATE reduces the datasets to between 0.7% and 12.9% of the original dataset volumes. DEFLATE reduces the traffic datasets to a mean of 11.2% of the original data volume, the air temperature dataset to 3.0% of the original volume, and the rainfall dataset to 0.9% of the original dataset volume. As discussed in Section 4.3, DEFLATE is a lossless compression algorithm that can reduce the data to a small form. However, the data must be decompressed before it can be used for analysis. In comparison,  $k$ D-STR allows the data to be retrieved without decompressing the entire reduced dataset. Therefore, DEFLATE is used as an indicator of the potential reduction in storage that is achievable by lossless compression algorithms, rather than as a directly comparable reduction method.

PCA, adapted for spatio-temporal data [43], reduces the traffic datasets to 11% of the original data volume when 1 principle component (PC) is stored per feature, 22% when 2 PCs are stored, and 33% when 3 PCs are stored. A mean NRMSE of 6.9%, 4.8% and 2.9% is achieved respectively. On the air temperature and rainfall datasets, PCA reduces each dataset to 25% of its original storage volume while incurring 0% NRMSE. In such cases, the number of principle components stored is equal to the number of features in the input

<sup>1</sup>A comparison for the other modelling techniques used by  $k$ D-STR can be inferred from Figure 4.5.

Table 4.3: Storage used by the reduced datasets after being compressed and reduced using existing techniques (values shown in MB). Results are shown for  $k$ D-STR using DCT-P modelling with  $\alpha = 0.1$  (high storage, low error) and  $\alpha = 0.9$  (low storage, high error).

Method	Road Traffic		Air Temperature		Rainfall	
	Min	Max	Min	Max	Min	Max
<i>Raw data</i>	3.3	23.6	10.4	11.7	8.2	9.2
DEFLATE	0.3	2.9	0.2	0.6	0.1	0.1
PCA (1)	0.4	2.6	1.2	1.3	0.9	1.1
PCA (2)	0.7	5.2	2.3	2.6	1.9	2.0
PCA (3)	1.1	7.8	3.4	3.9	2.7	3.1
IDEALEM	1.3	10.9	1.5	1.7	0.2	0.5
DCT-P ( $\alpha = 0.1$ )	0.7	69.3	0.1	25.8	0.1	11.9
DCT-P ( $\alpha = 0.9$ )	0.1	32.3	0.1	5.2	0.1	3.1

datasets (excluding spatial and temporal referencing features), thus no error is incurred.

IDEALEM is found to reduce the datasets to between 1.24% and 47.0% of the original data volume. For the traffic datasets, IDEALEM reduces the data to a mean of 42.3% of the original data volume and incurs a mean NRMSE of 17.8%. For the air temperature dataset, IDEALEM reduces the data to a mean of 12.1% of the original data volume and incurs a mean NRMSE of 11.5%. Finally, IDEALEM reduces the rainfall dataset to a mean of 2.4% of the original data volume while incurring a mean NRMSE of 21.1%.

In comparison to these results, when the parameter  $\alpha = 0.1$  and DTR-P modelling is used,  $k$ D-STR reduces the traffic datasets to a mean storage volume of 11.1% while incurring a mean NRMSE of 3.5%. Similarly, the air temperature dataset is reduced to a mean storage volume of 29.8% while incurring a mean NRMSE of 3.2%, and the rainfall dataset is reduced to a mean storage volume of 7.4% while incurring a NRMSE of 1.0%. This value of  $\alpha$  indicates a preference for minimising NRMSE at the cost of increases storage used. When  $\alpha = 0.9$ , a value which indicates a preference for minimising the storage used,  $k$ D-STR reduces the traffic datasets to a mean storage volume of 0.001% while incurring a mean NRMSE of 10.4%. Similarly, the air temperature dataset is reduced to a mean of 0.1% of the original storage volume while incurring a NRMSE of 9.3%, and the rainfall dataset to 0.01% while incurring a NRMSE of 2.3%.

Thus, when  $\alpha = 0.9$ ,  $k$ D-STR is found to reduce the datasets to a smaller storage volume than DEFLATE, PCA and IDEALEM. When  $\alpha = 0.1$ ,  $k$ D-STR reduces the rainfall dataset to a smaller storage volume than PCA, and the traffic datasets to a smaller storage volume than PCA and IDEALEM in the majority of cases. For all three datasets, the NRMSE incurred is significantly

lower than the NRMSE incurred by IDEALEM when  $\alpha = 0.1$ . These results indicate that  $k$ D-STR is able to reduce the datasets to smaller storage volumes than PCA and IDEALEM, often with significant improvements. Similarly,  $k$ D-STR is able to reduce the datasets while incurring a lower NRMSE than IDEALEM and, in the case of the traffic dataset, PCA as well.

#### 4.4.5 Choice of Spatial Referencing System

In some cases, datasets can be referenced using multiple spatial referencing systems (SRS). For example, the traffic datasets can be referenced using a 1-dimensional linear referencing system or a 2-dimensional Euclidean referencing system. When using a 1D SRS, each partition is referenced in space by a single beginning and ending value compared to multiple coordinates in a 2D SRS. Therefore, the quantity of storage used to store each partition is reduced. However, in a 1D SRS, discontinuities in the spatial domain force multiple partitions to be created that may be a single partition in a 2D SRS. This is seen in the traffic datasets, where the 1D SRS forces slip roads, which have a lower volume of traffic and therefore belong to different clusters, to break up the main carriageway partitions. In the 2D SRS, the main carriageway partitions *wrap around* the slip road partitions, and are therefore not broken into multiple partitions.

A comparison of the NRMSE incurred, storage used and number of partitions output by  $k$ D-STR when reducing the traffic datasets can be seen in Figure 4.8. In all cases, the NRMSE incurred and storage used for the 1D SRS is lower or approximately the same as the 2D SRS. For example, when using PLR-P modelling and  $\alpha = 0.1$ , the mean NRMSE incurred is 5.58% for the 1D SRS and 5.98% for the 2D SRS, while the storage used is 3.1% and 14.2% respectively. For PLR-P, PLR-C and DCT-C modelling, the storage used when  $\alpha = 0.1$  is notably lower when a 1D SRS is used compared to a 2D SRS.

The lower NRMSE incurred can be attributed to a higher number of partitions being stored for the 1D SRS compared to the 2D SRS (a comparison of the number of partitions output can be seen in Appendix A.5). In the case of PLR-P modelling when  $\alpha = 0.1$ , a mean of 11,997 partitions are output for the 1D SRS and 8,694 partitions are output for the 2D SRS. The higher number of partitions is attributable to the difference in slip roads and main carriageway breaking large partitions into multiple smaller partitions when the 1D SRS is used. However, the lower storage requirement of each partition does not increase the overall storage used by the reduced dataset, yielding a more efficient reduction with lower NRMSE as seen in Figure 4.8.

Therefore, the storage used by the reduced dataset may be decreased by using a SRS with fewer dimensions. Furthermore, if reducing the number of



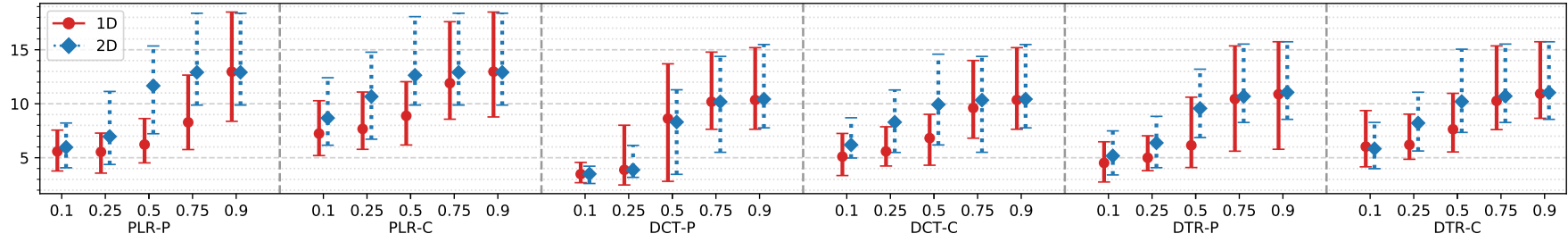
spatial dimensions increases the number of partitions, a lower NRMSE may be achieved. However, this is dependent on the spatial variability of the data and the dispersion of partitions in space.

#### 4.4.6 Impact of Input Dataset Size

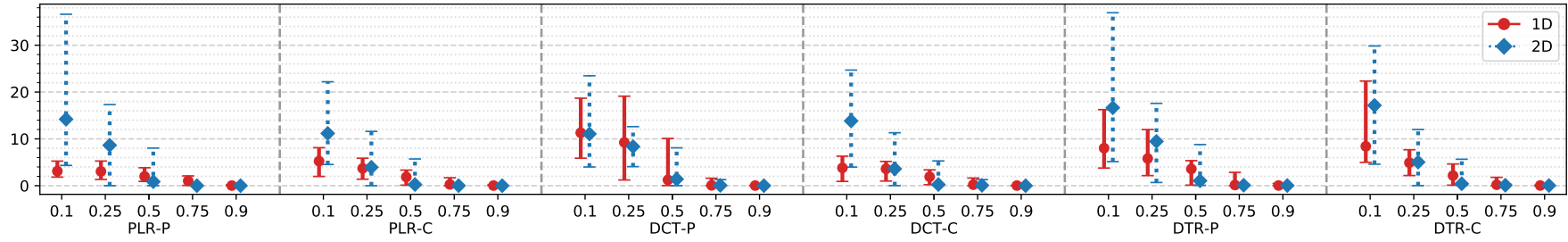
When  $\alpha = 0.1$ , increasing the size of the air temperature dataset in the temporal domain increases the percentage of relative storage used in some cases, and decreases the percentage used in others. That is, the percentage of storage used compared to the raw dataset (Equation 4.7) can increase or decrease. When the quantity of data increases, the number of partitions in the reduced dataset is proportionally higher and the NRMSE of the reduced data is lower. Similarly, when the percentage of data stored decreases, the number of partitions is proportionally lower and the NRMSE is higher. However, increasing the size of the air temperature dataset in the spatial domain decreases the storage used in all cases, and the NRMSE remains approximately the same. Furthermore, increasing the quantity of data in either domain decreases the storage used for the rainfall dataset and maintains approximately the same storage used, or lower, for the traffic datasets. Again, the NRMSE observed remains approximately the same. This can be seen in Figures 4.9 and 4.10, which show the effect of increasing the dataset size in the temporal and spatial domains respectively. Results are shown for DCT modelling, and similar results are observable for PLR and DTR modelling also (shown in Appendix A.6).

When  $\alpha \in \{0.75, 0.9\}$ , increasing the size of the dataset reduces the NRMSE in many cases and maintains approximately the same NRMSE in others. In most cases, the percentage of storage used relative to the raw data remains approximately the same as only 1 partition is stored. However, in some cases when  $\alpha = 0.75$ , the number of partitions output is greater than 1 for smaller dataset sizes. Thus, as the size of the dataset increases, the number of partitions output decreases to 1, resulting in a decrease in storage used and an increase in NRMSE. When  $\alpha = \{0.25, 0.5\}$ , increasing the quantity of data in the dataset either leads to an increase in the storage used and subsequent decrease in NRMSE, or a decrease in storage and increase in NRMSE.

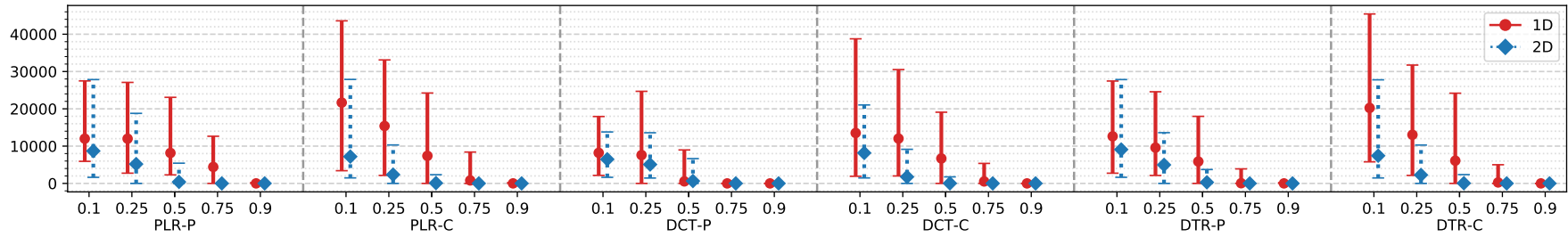
Increasing the quantity of data in the temporal domain leads to an approximately linear increase in the number of partitions output. This is demonstrated in Table 4.4 for DCT modelling, where increasing the quantity of air temperature data from  $0.25\times$  to  $1.0\times$  increases the number of partitions from 2,503 to 9,618 (PLR and DTR modelling results are shown in Appendix A.6). However, increasing the quantity of data in the spatial domain leads to a sub-linear increase in the number of partitions output. This is attributed to the lower spatio-temporal variance in the spatial domain requiring fewer partitions.



(a) NRMSE



(b) Storage Used



(c) Number of Partitions

Figure 4.8: NRMSE, storage and partitions output by  $k$ D-STR when reducing the traffic datasets referenced by 1D and 2D spatial referencing systems.

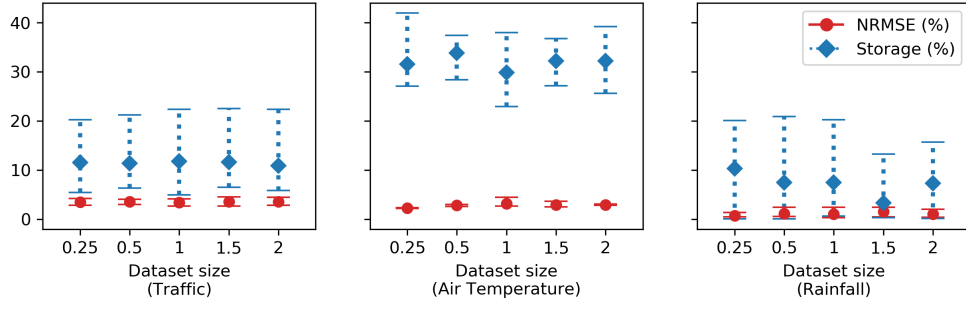


Figure 4.9: Effect on NRMSE and storage used as the quantity of data in the temporal domain increases, with results shown for DCT modelling when  $\alpha = 0.1$ .

#### 4.4.7 Running Time

When  $\alpha = 0.1$ , the running time required by the  $k$ D-STR algorithm ranged from 116 minutes to 12,108 minutes for the traffic datasets. For the air temperature dataset, the running time ranged from 611 minutes to 5,278 minutes, and for the rainfall dataset the running time ranged from 335 minutes to 5,399 minutes. In comparison, when  $\alpha = 0.9$ , the running time ranged from 4 to 9,642 minutes for the traffic datasets, 32 to 2,990 minutes for the air temperature dataset, and 17 minutes to 1,114 minutes for the rainfall dataset. However, these values capture the running time of unoptimised experimental Python code using a single processing core. Speedups of orders of magnitude may be achieved using optimised code written in efficient, compiled languages.

### 4.5 Discussion

The results presented in this chapter demonstrate the effectiveness of  $k$ D-STR at reducing spatio-temporal datasets with different characteristics to a set of partitions and models. These partitions and models require less storage than the original dataset, while incurring a NRMSE that is acceptable in many applications. These results also provide several conclusions that are enumerated in this section.

First, the parameter  $\alpha$  is shown to be effective at allowing the user to decide between minimising the NRMSE incurred and minimising the storage used by the reduced dataset. That is, for all datasets tested, the NRMSE incurred is lowest and storage used highest when  $\alpha$  is close to 0, while the NRMSE incurred is highest and storage used lowest when  $\alpha$  is close to 1. This is shown for a variety of datasets that exhibit different spatio-temporal characteristics, including different rates of variance in space and time, and different rates of continuous variability and sudden changes (events). The traffic dataset, which has high variability in space and time with discontinuities

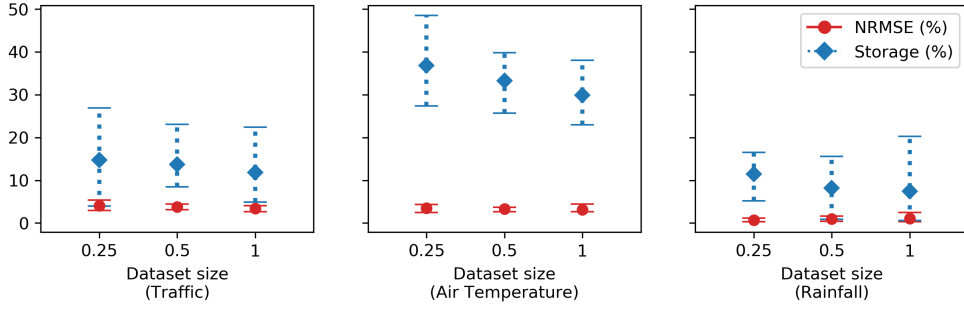


Figure 4.10: Effect on NRMSE and storage used as the quantity of data in the spatial domain increases, with results shown for DCT modelling when  $\alpha = 0.1$ .

Table 4.4: Average number of partitions stored as the quantity of data increases in the spatial and temporal domains. Results are shown for DCT modelling when  $\alpha = 0.1$  (high storage, low error).

	Temporal Domain					Spatial Domain		
	0.25×	0.5×	1×	1.5×	2×	0.25×	0.5×	1×
Traffic	1662	3676	7494	10344	12881	2736	4300	7494
Air Temp.	2503	5569	9618	15468	21247	3190	5499	9618
Rainfall	848	1188	2448	1632	5092	1078	1429	2448

in the spatial domain, incurs the highest mean NRMSE for all values of  $\alpha$  of the three datasets tested. In contrast, the rainfall dataset, which contains large areas and time periods of the same feature value (i.e. 0 mm rainfall), incurs the lowest NRMSE and requires the lowest quantity of storage to store the reduced dataset. The presence of spatial discontinuities is identified by the  $k$ D-STR partitioning technique, and the number of partitions created in the spatial and temporal domains is correlated with the variability of the datasets in the spatial and temporal domains. Overall, these findings support Hypothesis H1.1 and demonstrate the generality of spatio-temporal datasets that can be reduced by  $k$ D-STR.

Second, DCT and DTR incur lower NRMSE than PLR modelling for all of the datasets tested, although in many cases the storage used is greater. For the air temperature and rainfall datasets, modelling on clusters uses less storage than modelling on partitions. However, this is correlated with fewer partitions being stored, and a higher NRMSE. Furthermore, for the traffic dataset, modelling on clusters uses more storage while also incurring a higher NRMSE. These results contradict Hypothesis H1.2, which states that modelling on clusters would be more efficient when more than 1 partition is output. In particular, for the traffic datasets when DCT modelling is used and  $\alpha = 0.1$ , the number of partitions output when modelling on clusters is higher than the number output when modelling on partitions, yet the mean NRMSE is higher.

In this case, the number of model coefficients needed to reduce the NRMSE is greater than the number permitted by the heuristic function.

However, since increasing the number of partitions leads to larger improvements in the heuristic value per step than increasing the number of model coefficients, the greedy  $k$ D-STR algorithm prefers to maximise the number of partitions in the reduced dataset rather than maximise the number of model coefficients. Thus, a less greedy adaptation of  $k$ D-STR may be able to increase the number of coefficients rather than increase the number of partitions stored. This may result in lower NRMSE and less storage used when modelling on clusters versus modelling on partitions. However, adaptations of the algorithm and further testing are required to confirm this hypothesis.

Third,  $k$ D-STR is shown to yield a more efficient reduction when a 1D spatial referencing system (SRS) is used versus a 2D SRS. For the traffic datasets,  $k$ D-STR consistently incurs a lower or similar NRMSE while using less or approximately equal storage when a 1D SRS is used. A similar relationship between  $\alpha$  and the NRMSE and storage used is found. However, the fewer coordinate values required to store the spatial bounds of the partitions in 1D space results in less storage used by the reduced dataset. This confirms Hypothesis H1.3, and further demonstrates the generality of spatio-temporal datasets that can be reduced by  $k$ D-STR.

Fourth, increasing the dataset size while maintaining approximately the same spatio-temporal variance and density of instances is shown to increase the size of the reduced dataset. However, when  $\alpha = 0.1$ , increasing the size of the dataset can lead to a decrease in storage ratio while the NRMSE remains approximately the same. Conversely, when  $\alpha = 0.9$ , increasing the dataset size can decrease the NRMSE while maintaining approximately the same storage ratio. For some results presented in Section 4.4.6, the percentage of storage used increases as the dataset size increases, although this is attributable to an increase in the number of partitions stored as a result of increased spatio-temporal variance in the data. These results therefore support Hypothesis H1.4.

Finally,  $k$ D-STR is shown to achieve similar storage ratios to DEFLATE when  $\alpha = 0.9$ , and smaller storage ratios than PCA and IDEALEM for all three datasets. When  $\alpha = 0.9$ ,  $k$ D-STR achieves a smaller or comparable NRMSE for the traffic and rainfall datasets compared to PCA and IDEALEM. When  $\alpha = 0.1$ ,  $k$ D-STR achieves a lower NRMSE than IDEALEM for all three datasets, and a lower NRMSE than PCA for the traffic dataset. While DEFLATE is able to compress the data to a smaller form than the input dataset, the entire dataset must be decompressed before the data can be used analysed. Thus, the user must have sufficient memory to store the decompressed raw data and cannot use the compressed dataset directly. In comparison,  $k$ D-STR

allows the source data to be reconstructed by inputting the desired locations and times into the model(s) of the relevant partitions in the reduced dataset. Compared to PCA,  $k$ D-STR is able to model the datasets without requiring data be stored for each instance individually, allowing  $k$ D-STR to reduce the data to an even smaller form. Furthermore, compared to IDEALEM,  $k$ D-STR takes advantage of the spatio-temporal variability of the data to provide a more informed partitioning of the dataset without requiring the user to set a window size. This allows  $k$ D-STR to achieve smaller storage ratios in some cases and lower NRMSE in others. Thus,  $k$ D-STR is able to reduce datasets to a smaller form than the state-of-the-art whilst incurring a lower NRMSE in some cases, or a comparable NRMSE and storage ratio in others. In addition, the user is able to indicate their preference for minimised storage or minimised NRMSE by using an appropriate value for the parameter  $\alpha$ .

## 4.6 Summary

In this chapter a novel algorithm,  $k$ D-STR, for reducing spatio-temporal datasets has been presented. Using a novel partitioning method,  $k$ D-STR partitions the data in the spatial and temporal domains using the variability of the data, rather than the fixed-size partitioning scheme used by IDEALEM and ISABELA. By fitting a model to the data within each partition,  $k$ D-STR is able to store the data in a reduced form while minimising the NRMSE incurred. Compared to IDEALEM,  $k$ D-STR is often able to reduce the data to a smaller form. Furthermore,  $k$ D-STR allows for direct retrieval of the data without requiring multiple instances be retrieved and post-processed, as is required by ISABELA. These properties allow  $k$ D-STR to reduce datasets to a small form while permitting direct retrieval of the data. In Chapter 5, the speedup in processing times for analysis achieved by reducing the datasets with  $k$ D-STR is demonstrated empirically.

## Chapter 5

# Augmenting and Reducing Spatio-Temporal Datasets

Often, data scientists wish to analyse multiple spatio-temporal datasets in the context of each other. In Chapter 2, commonly used methods for linking datasets were presented that augment a *primary* dataset with information from one or more *supplementary* datasets. These methods select or impute instances from each supplementary dataset at locations and times that are relevant to the instances in the primary dataset, and append onto each primary instance features calculated using these supplementary instances.

However, the volume of spatio-temporal datasets can limit the number of primary and supplementary instances that can be processed in reasonable time. In an example described later in this chapter (see Section 5.4), augmenting a road condition dataset with road traffic and weather information takes approximately 4 hours using raw data. This length of time may be prohibitive for some applications, and the time taken can grow superlinearly as the size of the datasets increases. In Chapter 4, the *kD-STR* method was introduced for reducing the volume of such datasets while allowing for fast reconstruction and imputation of the data. Thus, *kD-STR* can be used to reduce the supplementary datasets and reduce the quantity of data that needs to be processed during linking. The *kD-STR* algorithm uses the RES heuristic to minimise the difference between the original (input) instances and the reconstructed instances. However, in the context of linking datasets, this may lead to unnecessary information retention in the reduced supplementary dataset. That is, *kD-STR* may retain unnecessary spatial or temporal resolution, and may minimise the error introduced at locations and times that are unlikely to be linked to the primary instances. For example, consider a scenario in which the rainfall dataset is used to calculate the total rainfall that fell on a set of farmers' fields over 1 year. Rather than minimising the difference between each original rainfall instance and the same instances after they have been

reconstructed from the reduced dataset, the difference between the total yearly rainfall value calculated for each farmers' field should be minimised. This approach concentrates supplementary information retention in the areas and times applicable to the primary dataset, and minimises the error introduced to the linking process by the reduction process.

Therefore, this chapter introduces a wrapper method for integrating  $k$ D-STR with the dataset linking process. First, an alternative heuristic for  $k$ D-STR is presented for use in the context of data augmentation. This alternative heuristic, namely the *Linked Error and Storage* (LES) heuristic, represents the main contribution of this chapter, and prioritises information retention in the areas and time periods most applicable to the primary dataset. In doing so, the LES heuristic yields a reduced dataset that is at least as accurate as the same input dataset reduced using the RES heuristic presented in Chapter 4, whilst using at most the same quantity of data. Second, the reduction and augmenting processes are combined for outputting a reduced augmented dataset and for speeding up the linking process. Finally, this chapter presents an empirical case study that demonstrates the improvement achieved using the LES heuristic. In this case study, a road condition dataset is augmented using aggregated forms of the traffic, air temperature and rainfall datasets presented in Chapter 3, as well as the same datasets after they have been reduced using  $k$ D-STR with the RES heuristic and the LES heuristic presented in this chapter.

This chapter is organised as follows. In Section 5.1, the process of augmenting one dataset with information from one or more other datasets is explained. In Section 5.2, the LES heuristic for the  $k$ D-STR algorithm is presented for reducing the supplementary datasets used to augment a primary dataset. Furthermore, Section 5.3 discusses how the  $k$ D-STR algorithm with LES heuristic can be integrated into the wider augmentation process. Section 5.4 describes the experimental setup used to evaluate the LES  $k$ D-STR heuristic, and Section 5.5 explains the results of this evaluation. In Section 5.6, the consequences of these results are discussed. Finally, Section 5.7 provides concluding remarks for this chapter.

## 5.1 Augmenting Primary Datasets with Supplementary Datasets

Often, data analysts wish to augment a primary dataset  $D^P$  with information from  $n$  supplementary datasets,  $D^1, \dots, D^n$ . For example,  $D^P$  may be weekly sales data of de-icer at stores across a country, with each instance representing one store per week. Furthermore,  $D^1$  may be hourly air temperature and



rainfall observations recorded at weather stations,  $D^2$  may be hourly footfall counts of people walking past sensors near to the stores,  $D^3$  may be sales data for related products, and  $D^4$  may be vehicle ownership data. In this scenario, each sales instance is augmented with the minimum temperature and total precipitation observed in the week before and after the sales instance was recorded, as well as the number of related products sold in the same time frame and the number of vehicles owned within the local population. The result of augmenting  $D^P$  with  $D^1, \dots, D^n$  is the dataset  $D^{\text{Aug}}$ .

The linking procedure for raw datasets (i.e. datasets that have not been reduced), is shown in Algorithm 2. Here, a subset of zero or more instances  $L^l \subset D^l$  is chosen from each supplementary dataset  $D^l$ , where  $1 \leq l \leq n$ . The subset is chosen by the function  $\text{link}()$  for each instance  $d_{s,t} \in D^P$ , and consists of one or more instances  $d_{u,v} \in D^l$ . In our de-icer sales example,  $\text{link}()$  selects all weather instances in  $D^1$  from the nearest weather station to location  $s$  in the week before and after time  $t$ , as well as footfall instances in  $D^2$  from the nearest footfall sensor in the same time period, related sales instances from the same store in the same time period, and the number of vehicles owned in the surrounding area. Then, features are engineered by the function  $\text{features}()$  using  $L^1, \dots, L^n$ , and the engineered features appended onto  $d_{s,t}$ . This results in the augmented instance  $d'_{s,t} \in D^{\text{Aug}}$ .

---

**Algorithm 2:** Augmenting primary dataset  $D^P$  with information from multiple supplementary datasets

---

**Data:** Primary dataset  $D^P$  and supplementary datasets  $D^1, \dots, D^n$

**Result:**  $D^{\text{Aug}}$ , the resulting dataset after  $D^P$  is augmented with  $D^1, \dots, D^n$

---

```

1  $D^{\text{Aug}} \leftarrow \emptyset$ 
2 for  $d_{s,t} \in D^P$  do
3   for  $1 \leq l \leq n$  do
4      $L^l \leftarrow \text{link}(d_{s,t}, D^l)$ 
5   end
6    $d'_{s,t} \leftarrow d_{s,t} + \text{features}(d_{s,t}, L^1, \dots, L^n)$ 
7    $D^{\text{Aug}} \leftarrow D^{\text{Aug}} + d'_{s,t}$ 
8 end
```

---

### 5.1.1 Methods for Imputing Supplementary Instances

In some cases, only one time and location in  $D^l$  are applicable to each primary instance, thus  $L^l$  contains one instance. In other cases, multiple times or locations are applicable, and  $\text{features}()$  aggregates the selected instances — for the de-icer sales example,  $L^1$  will contain up to 336 hourly temperature instances. However,  $D^l$  may not contain instances at the exact times and

locations applicable to a primary instance  $d_{s,t} \in D^P$ . That is, it is unlikely that the weather dataset  $D^1$  contains instances at the exact location  $s$  of each store. Therefore, the instances in  $L^l$  are imputed using the instances that are present in  $D^l$ . In Chapter 2, examples found in literature were presented that use the  $k$  nearest neighbour (kNN)<sup>1</sup> and neighbourhood techniques. These techniques are commonly used because of their simplicity and sufficient accuracy in such scenarios. Both methods either average the selected instances or weight them according to their distance or perceived accuracy/relevance. In this section, these methods for creating the subset of instances  $L^l$  are discussed.

Given a primary instance  $d_{s,t} \in D^P$ , assume that we are interested in knowing the feature values of  $D^l$  at a location  $u$  and time  $v$ , denoted  $(u, v)$ . Note that  $(u, v)$  may not necessarily be the same as  $(s, t)$ ; for example, when predicting de-icer sales, the weather of the previous day may be applicable and so  $u = s$  and  $v = t - 1$  day<sup>2</sup>. However, when no instance exists at that location and time, i.e.  $d_{u,v} \notin D^l$ , the nearest neighbours to  $(u, v)$  in  $D^l$  can be used to impute  $\tilde{d}_{u,v}$ . That is, the spatio-temporal distance between  $(u, v)$  and  $(w, x)$  can be calculated for each instance  $d_{w,x} \in D^l$ , and the instances with the lowest spatio-temporal distance used to estimate the feature values of  $D^l$  at  $(u, v)$ .

Let the function  $\delta_{\text{ST}}(u, v, w, x)$  be a spatio-temporal distance metric between the desired instance being estimated  $\tilde{d}_{u,v}$  and any instance  $d_{w,x} \in D^l$ . The function combines both the spatial distance metric  $\delta_{\text{S}}(u, w)$  and temporal distance metric  $\delta_{\text{T}}(v, x)$  into a single spatio-temporal distance metric, although it can be ambiguous how these two metrics should be combined. One method for combining the two metrics is to use the product of the two distances [144], however this ignores the possibility of one metric dominating the other owing to the units used. Instead, the two metrics can be weighted as shown in Equation 5.1 [7, 57]. Weighting the metrics instead of using their product also allows other metrics to be incorporated into the distance function, such as utility scores of sensor accuracy and precision, or metrics that consider the spatial and temporal restrictions inherent on the processes sampled by the supplementary datasets. Choosing a value for the scaling factor  $\mu$  is subjective and is explored in Section 5.1.2.

$$\delta_{\text{ST}}(u, v, w, x) = \delta_{\text{S}}(u, w) + \mu \cdot \delta_{\text{T}}(v, x) \quad (5.1)$$

Using the nearest spatio-temporal neighbour to estimate the feature values at a desired location and time is a special case of the  $k$  nearest neighbours

---

<sup>1</sup>Note that  $k$  here refers to the number of neighbour instances selected rather than the number of spatio-temporal dimensions.

<sup>2</sup>The choice of values for  $u$  and  $v$  given location  $s$  and time  $t$  are dataset specific and are calculated using domain knowledge.

algorithm (kNN). The nearest  $k$  instances to  $(u, v)$  in  $D^l$  are chosen and their feature values used to estimate the values of those features at  $(u, v)$ . Choosing multiple nearest neighbours (i.e.  $k > 1$ ) may result in a more accurate estimate than choosing the single nearest neighbour as it dilutes anomalous or noisy supplementary instances. However, choosing too many nearest neighbours can be detrimental as instances far from  $(u, v)$  can give a poor estimate of  $\tilde{d}_{u,v}$ . After the  $k$  nearest instances to  $(u, v)$  have been chosen from  $D^l$ , the estimated value  $\tilde{d}_{u,v}$  can be calculated using:

$$\tilde{d}_{u,v} = \frac{1}{k} \sum_{d_{w,x} \in A} d_{w,x} \quad (5.2)$$

where  $A \subseteq D^l$  is the subset of  $k$  nearest instances to  $(u, v)$ .

While kNN is simple to implement and often sufficiently accurate, it ignores the effect of spatio-temporal autocorrelation. To maintain spatio-temporal autocorrelation, the instances can be weighted by their distance to  $(u, v)$ , such that nearer instances are more heavily weighted. This is referred to as inverse distance weighted  $k$  nearest neighbours (IDW kNN), and the estimated value can be calculated using:

$$\tilde{d}_{u,v} = \frac{1}{\sum_{d_{w,x} \in A} w_{w,x}} \sum_{d_{w,x} \in A} w_{w,x} \cdot d_{w,x} \quad (5.3)$$

where,

$$w_{w,x} = \frac{1}{\delta_{\mathbb{T}}(u, v, w, x)}$$

However, while kNN and IDW kNN can be more accurate than the nearest neighbour method, choosing a value for  $k$  can be difficult. For example, when the density of supplementary instances varies over space and time, the proximity of supplementary instances can differ between primary instances. That is, for some primary instances, the  $k$  nearest supplementary instances may be near but for others the  $k$  nearest neighbours may be quite far. In some cases this may be desirable, however when it is not, it may be more beneficial to use a fixed spatio-temporal neighbourhood around  $(u, v)$ . In doing so, the user has to select a spatial limit  $v$  and temporal limit  $\tau$  as the maximum distance a supplementary instance can be from a primary instance. In this case, given a function  $neighbourhood(D^l, u, v, v, \tau)$  which returns the subset  $A \subseteq D^l$  of instances within the spatial limit  $v$  and the temporal limit  $\tau$ ,  $\tilde{d}_{u,v}$  can again be calculated using Equation 5.2. The neighbourhood method may also be adapted to use inverse distance weighting using Equation 5.3. A choice for  $v$  and  $\tau$  can be made using the techniques described in Section 5.1.2.

By investigating the accuracy of the imputation methods discussed above

using an approach similar to those presented in Section 2.1.2, an appropriate technique may be chosen. Gaining this domain specific knowledge may take time, however knowledge gained for one sample of data may be useful for future samples of the same data that exhibit the same distributions. This removes the need to repeat the knowledge gathering process. For example, knowledge gained when linking a sample of weather data may be useful when linking a later sample of weather data for the same area. Yet, for some supplementary datasets, using an average or weighted average of instances near to the desired location and time may not provide a sufficiently accurate estimate of the dataset’s feature values. One method for overcoming this issue is to fit a predictive model to the instances in  $D^l$  and impute the desired instances using the model. Such models may provide a more accurate estimate by capturing the patterns and trends present in the data, rather than interpolating between nearest instances. A predictive model may be as simple as a linear or polynomial regression model, using the locations and times of the supplementary instances as predictors and their feature values as responses. This approach is taken by  $k$ D-STR, although more sophisticated modelling techniques that lie outside the scope of this thesis may also be useful.

### 5.1.2 Choosing Parameters for Imputation Methods

By understanding the characteristics of the primary and supplementary datasets, such as using the techniques discussed in Chapter 3, an appropriate imputation method can be chosen. The aim is to choose a method that minimises the computation required whilst achieving sufficient accuracy for the given scenario. For example, if the maximum distance between any sensor in  $D^P$  and its nearest neighbour in  $D^l$  is known to be 7 miles, and experimental variograms show that  $D^l$  does not vary significantly over 7 miles, then using the nearest instance in time at the nearest sensor may be sufficiently accurate. That is, using the kNN technique with  $k = 1$  is sufficient. However, if the experimental variograms show  $D^l$  varies significantly over 7 miles, we can expect an instance in  $D^l$  recorded 7 miles from the primary instance  $d_{s,t}$  to be too different to be applicable. In this case, increasing the value of  $k$  may be more accurate, despite being less computationally efficient.

As well as choosing an appropriate imputation method, a weighting  $\mu$  of the spatial and temporal distance metrics in the distance metric  $\delta_{\text{ST}}(u, v, w, x)$  is also required. Using experimental variograms, a value for  $\mu$  can be found that produces similar differences between instances over space and time. For example, if a distance of 20 miles produces an average difference of  $1^\circ\text{C}$  in an air temperature dataset, and 1 hour produces a similar difference of  $1^\circ\text{C}$ , 20 miles can be considered to be approximately equal to 1 hour in the distance function.

This estimate is simple to infer and can be useful for datasets that exhibit gradual changes to feature values over space and time. However, it assumes that the variation of the supplementary datasets is approximately uniform across space and time. Other techniques that are computationally expensive but may be useful have been discussed in literature, such as iteratively optimising  $\mu$  by minimising the  $R^2$  or AIC values when combined with leave-one-out cross validation [64].

A final factor to consider when choosing an appropriate selection or imputation technique is the robustness of these techniques and their susceptibility to the addition of noise. A more in-depth discussion on analysing the robustness of techniques to such error is presented in Chapter 6. Once an imputation method and its parameters have been chosen, these choices may also be appropriate for other samples of similar data that exhibit similar distributions, such as future samples of the same dataset. In such cases there may be no need to complete these investigations again.

## 5.2 Reducing Supplementary Datasets

To speed up the linking process of augmenting  $D^P$  with the supplementary datasets  $D^1, \dots, D^n$ , the supplementary datasets can be reduced beforehand. In doing so, each supplementary dataset  $D^l$  is reduced to a set of partitions and models  $\langle P^l, M^l \rangle$ , with each model linked to one or more partitions (see Chapter 4.2.2). Then, instead of calculating the distance between each location and time  $(u, v)$  and each instance  $d_{w,x} \in D^l$ , the feature values can be retrieved directly from the reduced dataset. That is,  $(u, v)$  has to be compared with the spatio-temporal bounds of each partition in the reduced dataset  $\langle P^l, M^l \rangle$ , and the model from the selected partition used. Since the number of partitions in the reduced dataset is smaller than the number of instances in  $D^l$ , i.e.  $|P^l| \ll |D^l|$ , the linking process is faster<sup>3</sup>.

However, when reducing the supplementary datasets,  $kD$ -STR will attempt to capture the variation of the data in areas and time periods where the feature values vary the most. Yet, these may be applicable to few primary instances. Instead, it is beneficial to consider both the distribution of the areas and time periods applicable to the primary instances and the variation of the supplementary instances over space and time when prioritising model accuracy. That is, rather than minimising the error in the reconstructed supplementary datasets, the reduction process should minimise the error introduced in the features engineered during the linking process. To achieve this, the RES

---

<sup>3</sup>This assumes that the computation required to compare  $(u, v)$  with each instance  $d_{w,x} \in D^l$  is approximately equal to the time required to compare  $(u, v)$  with each instance  $p_i^l \in P^l$ .

heuristic presented in Chapter 4 (Equation 4.8) can be adapted to consider the error of the features engineered using  $D^l$ , thus forming the LES heuristic:

$$h(D_*^P, D^l, \langle P^l, M^l \rangle) = \alpha \cdot q(D^l, \langle P^l, M^l \rangle) + (1 - \alpha) \cdot e(D_*^P, \langle P^l, M^l \rangle) \quad (5.4)$$

Here,  $q(D^l, \langle P^l, M^l \rangle)$  measures the ratio between the volume of storage required to store the reduced supplementary dataset and that required to store the raw supplementary dataset. Furthermore,  $e(D_*^P, \langle P^l, M^l \rangle)$  measures the difference between the features engineered during the linking process using the raw supplementary dataset  $D^l$  versus those engineered using the reduced supplementary dataset  $\langle P^l, M^l \rangle$ . In order to calculate the error between the features engineered using  $D^l$  and  $\langle P^l, M^l \rangle$ , the linking process must be completed for each primary instance. That is, we have to augment each primary instance using the raw dataset  $D^l$ , and the reduced dataset  $\langle P^l, M^l \rangle$ , and calculate the difference in the engineered feature values. This presents an issue, as the linking procedure being sped up has to be performed to calculate the error metric of the reduction procedure. In the example presented later in Section 5.4, this linking process takes approximately 4 hours and has to be completed to calculate the heuristic value at least twice per iteration of the reduction algorithm. Completing this process within the reduction process is therefore infeasible. Instead, a representative sample of the of the primary dataset  $D_*^P \subset D^P$  is used to estimate the error introduced by reducing the supplementary datasets.

The functions used in Equation 5.4 are defined in Equations 5.5 and 5.6. In Equation 5.6, the sample  $D_*^P \subset D^P$  is augmented using the raw supplementary dataset to obtain  $D_*^{\text{Aug}}$ . Furthermore, an approximation  $\hat{D}_*^{\text{Aug}}$  is obtained by augmenting  $D_*^P$  using the reduced supplementary dataset. Note that  $F_{\text{Aug}}^l$  is the set of features engineered during the linking process using dataset  $D^l$ .

$$q(D^l, \langle P^l, M^l \rangle) = \frac{\text{storage}(\langle P^l, M^l \rangle)}{\text{storage}(D^l)} \quad (5.5)$$

$$e_{\text{NRMSE}}(D_*^P, \langle P^l, M^l \rangle) = \frac{1}{f} \sum_{f \in F_{\text{Aug}}^l} \frac{\psi(D_*^{\text{Aug}}, \hat{D}_*^{\text{Aug}}, f)}{\text{range}(f)} \quad (5.6)$$

$$\psi(D_*^{\text{Aug}}, \hat{D}_*^{\text{Aug}}, f) = \sqrt{\frac{\sum_{d_{s,t} \in D_*^{\text{Aug}}} (d_{s,t}^f - \hat{d}_{s,t}^f)^2}{|D|}}$$

### 5.2.1 Sampling the Primary Dataset

A naive method for creating the subset  $D_*^P \subset D^P$  is to select a fixed number of instances at random (without replacement) with equal probability. However, this may lead to a sample that is not representative of the distribution of instances in the spatial and temporal domains. Such a non-representative sample would focus model accuracy in areas and time periods that are not applicable to the majority of instances in  $D^P$ . Instead, a stratified sample should be used, and a selection of appropriate techniques have been discussed in literature [41]. For example, the spatial and temporal domains may be partitioned into non-overlapping strata and a number of instances chosen from each stratum that is proportional to the number of instances within that stratum. Such a process is referred to as *stratified random sampling*. The number of instances in a stratum  $j$  is defined as  $\lceil m \times \frac{|D_j^P|}{|D^P|} \rceil$ , where  $m$  is the fraction of instances to be sampled,  $|D^P|$  is the number of instances in the primary dataset, and  $|D_j^P|$  is the number of instances in stratum  $j$  from dataset  $D^P$ .

An alternative method is to choose a fraction  $m$  of instances from  $D^P$  at random while ensuring that each instance is at least a spatial distance  $\gamma_S$  and temporal distance  $\gamma_T$  apart from any other instance [31]. However, choosing values for  $\gamma_S$  and  $\gamma_T$  that are too high may under-sample more dense areas or time periods in  $D^P$ , and over-sample less dense areas/periods. Conversely, choosing values that are too low may over-sample more dense areas/periods and under-sample less dense areas/periods. Therefore, the stratified random sampling technique defined in [41] is used in this thesis owing to its use in literature and the ease of parameter value selection.

### 5.2.2 Preprocessing for Further Reduction

As well as reducing a supplementary dataset using  $kD$ -STR, the dataset can be preprocessed to remove redundant data prior to reduction. This has two benefits. First, the spatial and temporal resolutions of the primary dataset may be significantly higher or lower than the resolutions of the supplementary datasets. When a supplementary dataset has a higher resolution than the primary dataset,  $kD$ -STR will retain unnecessary resolution and fail to take advantage of the storage saving available. For the de-icer sales example presented in Section 5.1, consider a case in which the minimum distance between any two stores in  $D^P$  is 20 miles, and we wish to augment each sales instance  $d_{s,t} \in D^P$  with the daily mean air temperature recorded at the nearest sensor to  $s$  in the dataset  $D^1$ . Furthermore, the maximum distance between any pair of weather sensors in  $D^1$  is 1.5 miles, and each sensor records one instance per hour. Retaining the hourly temporal resolution and high spatial resolution

of  $D^1$  is unnecessary, thus reducing the resolution of the supplementary datasets when possible will further reduce the quantity of storage used and time required to link the datasets.

Second,  $k$ D-STR may retain information and model accuracy in areas or time periods that are not applicable to the primary dataset, failing to take advantage of the storage saving available. Again, removing instances outside the spatial and temporal bounds that are applicable to the primary dataset will further reduce the quantity of storage used, as well as allowing  $k$ D-STR to prioritise model accuracy in the areas and times most applicable to the primary dataset. For each supplementary dataset  $D^l$ , a spatial boundary  $S_{\text{bound}}^l = \langle (s_{1,b}, s_{1,e}), \dots, (s_{\mathcal{D},b}, s_{\mathcal{D},e}) \rangle$  of beginning and ending coordinates in each dimension, and a temporal boundary  $T_{\text{bound}}^l = (t_b, t_e)$  are computed. Then, all instances in  $D^l$  that are located outside of  $S_{\text{bound}}^l$  and  $T_{\text{bound}}^l$  are removed. Forming the boundaries requires knowledge of the instances in  $D^P$  and  $D^l$ , as well as knowledge of the furthest a supplementary instance in  $D^l$  can be to remain applicable to an instance in  $D^P$ . This knowledge can be computed using the maximum distance between the primary instances, as well as domain knowledge of the supplementary datasets aided by experimental variograms.

By using these two preprocessing steps prior to reducing the supplementary dataset with  $k$ D-STR and the LES heuristic, the linking process can be sped up further and the linked features engineered more accurately. A summary of reducing the supplementary datasets is shown in Algorithm 3. Here, lines 1–3 calculate the spatial and temporal resolutions of the primary dataset, and create a sample of the dataset that is representative of its distribution in space and time. Furthermore, lines 5–7 calculate the spatial and temporal bounds of  $D^l$  and remove instances from the dataset that are outside of these bounds. Finally, line 8 reduces the spatial and temporal resolutions of  $D^l$  given the resolution of  $D^P$ , and line 9 reduces  $D^l$  using  $k$ D-STR with the LES heuristic presented in Equation 5.4.

### 5.3 Integrating the Augmenting and Reduction Processes

Using the LES heuristic presented in Section 5.2, the supplementary datasets in a linking scenario can be reduced to speed up the linking process. However, in some scenarios we may also wish to reduce the augmented dataset  $D^{\text{Aug}}$ . In such cases, we may either: (i) perform the linking procedure using the raw primary dataset and either the raw supplementary datasets or reduced supplementary datasets, then reduce the linked output; or (ii) reduce the primary



---

**Algorithm 3:** Reducing the supplementary datasets  $D^1, \dots, D^n$  with  $k$ D-STR and the LES heuristic

---

**Data:** Primary dataset  $D^P$  and supplementary datasets  $D^1, \dots, D^n$   
**Result:** The reduced datasets  $\langle P^1, M^1 \rangle, \dots, \langle P^n, M^n \rangle$

```

1  $r_s \leftarrow \text{spatialRes}(D^P)$ 
2  $t_s \leftarrow \text{temporalRes}(D^P)$ 
3  $D_*^P \leftarrow \text{sample}(D^P, r_s, r_t)$ 
4 for  $1 \leq l \leq n$  do
5    $S_{\text{bound}}^l \leftarrow \text{spatialBoundary}(D^l, D^P)$ 
6    $T_{\text{bound}}^l \leftarrow \text{temporalBoundary}(D^l, D^P)$ 
7    $D^l \leftarrow \text{removeOutside}(D^l, S_{\text{bound}}^l, T_{\text{bound}}^l)$ 
8    $D^l \leftarrow \text{reduceRes}(D^l, r_s, r_t)$ 
9    $\langle P^l, M^l \rangle \leftarrow \text{lesKDSTR}(D^l, D_*^P)$ 
10 end

```

---

and supplementary datasets independently and output a set of partitions and models for each of the datasets, as well as a function for mapping between the partitions in each of the reduced datasets. These two cases can be referred to as *linking then reducing* and *reducing then linking* respectively.

After linking then reducing, linked instances can be imputed by inputting the desired locations and times into the reduced dataset models. To retrieve a single linked instance then requires  $\mathcal{O}(|P^{\text{Aug}}|)$  time, where  $|P^{\text{Aug}}|$  is the number of partitions in the reduced augmented dataset, as each partition in the reduced dataset has to be compared against the desired location and time. For reducing then linking, retrieving a single linked instance requires  $\mathcal{O}(n|P^{\text{max}}||P^P|)$  time. Here,  $n$  is the number of supplementary datasets, and  $P^{\text{max}}$  is the maximum number of partitions in any of the reduced datasets, i.e.  $|P^{\text{max}}| = \max_l |P^l|$ . Furthermore,  $P^P$  is the set of partitions in the reduced primary dataset. Though this requires more computation than retrieving an instance from the reduced augmented dataset, it may still be more efficient than linking the raw primary and supplementary datasets to retrieve an augmented instance.

Overall, the integrated process of reducing and linking a primary dataset with  $n$  supplementary datasets is as follows:

1. **Prepare the primary dataset:** choose whether to use the primary dataset  $D^P$  or the reduced primary dataset  $\langle P^P, M^P \rangle$ ; if required, reduce  $D^P$  using  $k$ D-STR
2. **Prepare the supplementary datasets:** for each supplementary dataset  $D^l$ , remove unnecessary data and reduce using  $k$ D-STR with the LES heuristic (Equation 5.4), as shown in Algorithm 3
3. **Initialise the linked dataset:**  $D^{\text{Aug}} \leftarrow \emptyset$  or  $P^{\text{Aug}} \leftarrow \emptyset$

4. **Augment the primary instances or partitions** (either  $d_{s,t} \in D^P$  or  $\hat{d}_{s,t}$  imputed from  $\langle P^P, M^P \rangle$ ):
  - (a) For each of the supplementary datasets (either  $D^l$  or  $\langle P^l, M^l \rangle$  for  $1 \leq l \leq n$ ), let  $L^l$  be the instances selected from  $D^l$  or imputed from  $\langle P^l, M^l \rangle$  that are applicable to  $d_{s,t}$  or  $p_i \in P^P$
  - (b) Append onto  $d_{s,t}$  features that are selected or engineered using  $L_1, \dots, L_n$ , or record links between  $p_i$  and  $L_1, \dots, L_n$
  - (c) Add the augmented instance  $d'_{s,t}$  to  $D^{\text{Aug}}$  or partition  $p_i$  to  $P^{\text{Aug}}$
5. **Output augmented dataset:**
  - If returning the augmented dataset in its reduced form, return  $D^{\text{Aug}}$ .
  - If returning the reduced dataset of  $D^{\text{Aug}}$ , reduce  $D^{\text{Aug}}$  using  $kD\text{-STR}$  and return the output partitions and models
  - If returning the reduced dataset of  $\langle P^{\text{Aug}}, M^{\text{Aug}} \rangle$ , return  $\langle P^{\text{Aug}}, M^{\text{Aug}} \rangle$  as well as the reduced supplementary datasets  $\langle P^1, M^1 \rangle, \dots, \langle P^n, M^n \rangle$  and the linking function between the datasets

## 5.4 Experimental Evaluation Methodology

To evaluate the impact of data reduction on the linking process, a case study is used that augments a road condition dataset using the road traffic, air temperature and rainfall datasets presented in Chapter 3. Research found in literature has shown that cracking, rutting and polishing of road surfaces are increased by surges in air temperature, precipitation and both the number and weight of vehicles passing over the road surface [110]. Thus, the road condition dataset contains features of cracking, rutting and polishing, and each instance is indexed by a location, two measurement dates and the daily change in condition feature values between these two dates<sup>4</sup>. Each instance is augmented with features calculated using the raw and reduced traffic, air temperature and rainfall datasets. Then, a comparison is made between the linked features output after augmenting the road condition dataset using the raw and reduced supplementary datasets. Furthermore, since the supplementary datasets may be reduced using  $kD\text{-STR}$  with the RES or LES heuristics, a comparison of the two heuristics is made.

More specifically, the following hypotheses are investigated in this case study:

---

<sup>4</sup>That is, the total change in cracking, rutting and polishing features divided by the number of days between the two surveys represented by an instance.

H2.1 *Compared to the RES heuristic, the LES heuristic yields more accurate linked features.*

That is, given approximately the same quantity of storage used, the features engineered using the reduced datasets that are reduced using the LES heuristic are more accurate than the datasets that are reduced using the RES heuristic. Conversely, to achieve approximately the same NRMSE, the LES heuristic requires less storage for the reduced dataset than the RES heuristic. Since the LES heuristic aims to minimise the error introduced in the linking process rather than the reconstruction of the original supplementary instances, this is the case. Furthermore, the baseline used to measure accuracy is the linked features engineered using the raw supplementary datasets.

H2.2 *Compared to the RES heuristic, the LES heuristic stores more model coefficients for those partitions nearest to the instances in  $D^P$ .*

That is, only those partitions which contain the spatial locations of instances in  $D^P$  and overlap with the time periods of each instance store more than 1 model coefficient. If a model is linked to partitions that are not used to augment any instances in  $D^P$ , improving the model's accuracy does not improve the error metric in the heuristic. However, improving a model that is used to augment many primary instances does. It therefore follows that partitions that are used to augment more primary instances are modelled using more coefficients than those that are far away.

H2.3 *Supplementary instances that are far from the primary instances are less accurately reconstructed after being reduced with the LES heuristic than the RES heuristic.*

This hypothesis builds on Hypothesis H2.2. When the original supplementary instances are reconstructed from the reduced dataset, those instances that are far from the locations and times of the primary instances are less accurately reconstructed than those that are close to the primary instances. Since those supplementary instances that are far from the primary instances are not used by the LES heuristic in the linking process, the partitions in which they reside are not be modelled as accurately as those that are used in the linking process.

Each of these hypotheses are evaluated using the datasets presented in Section 5.4.1 and metrics presented in Chapter 4.

### 5.4.1 Preprocessed Datasets

During the linking process, each primary instance in the road condition dataset is appended with the following features: (i) *mean daily minimum temperature*; (ii) *mean daily average temperature*; (iii) *mean daily maximum temperature*; (iv) *mean daily total precipitation*; (v) *mean daily total traffic flow*; (vi) *mean daily large vehicle count*. The input datasets used to create these features are the road traffic, air temperature and rainfall datasets presented in Chapter 3. However, only road condition data for the M1 motorway is available, and so only road traffic data for this road is considered. The supplementary datasets are first preprocessed using the process described in Section 5.2.2 to reduce the temporal resolution of the datasets to daily aggregates. The spatial resolution is not reduced as the road condition dataset has a higher spatial resolution than the supplementary datasets. However, instances that are far from the road condition instances are retained in all three datasets as this is required for testing hypotheses H2.2 and H2.3.

The properties of the datasets after preprocessing are as follows.

#### Road Condition Dataset (Primary)

The road condition dataset contains features about the rutting, skid resistance, texture and longitudinal profile of the M1 motorway [1]. Each instance is recorded at a specific time for a section of road up to 100 m in length on the main carriageway of the motorway. Each section is surveyed between 6 and 12 months apart, thus each instance can be linked to a previous instance recorded for the same section of road.

Before preprocessing, the dataset contains all instances recorded in 2016 and 2017. During preprocessing, each instance is first linked to the previous instance recorded at the same sensor and the features of the previous survey instance are appended onto the instance being considered. Then, all instances that are not linked to a previous instance are removed, yielding only instances for which two dates (a first survey and second survey) and two sets of features (those for each survey) are present. Thus, after preprocessing, each instance represents a period of time defined between a start and end date, and contains start and end values for each of the dataset's features. Furthermore, the mid-point in space of each instance is also determined to give each section of road a single point location in space. After preprocessing, the primary dataset contains 2719 instances between June 2016 and September 2017.

#### Road Traffic Dataset (Supplementary)

To form the road traffic dataset, data samples for the M1 motorway from the traffic dataset introduced in Chapter 3 are used. During preprocessing, each

sensor’s 15-minute feature values are aggregated into daily totals. Furthermore, only the *total daily vehicle count* and *total daily count of vehicles longer than 6.6 m* features are used as these have been shown to affect the features in the road condition dataset [110]. Only traffic sensors on the main carriageway of the road are considered, and those on exit and entry slip-roads to the motorway are omitted. This yields 155 sensors on the M1 motorway for each day in 2016 and 2017, giving a total of 69,715 instances. The raw road traffic data file is 6.1 MB in size. Figure 5.1a shows the distribution of feature values and experimental variogram for the road traffic dataset.

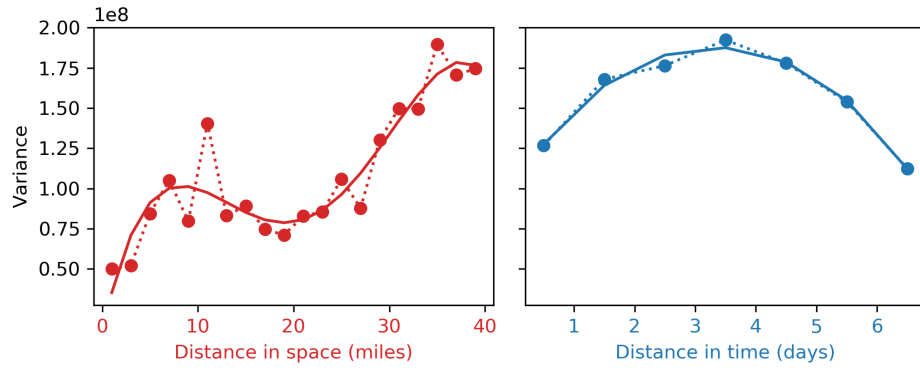
### **Air Temperature and Rainfall Datasets (Supplementary)**

The air temperature and rainfall datasets presented in Chapter 3 are used to provide temperature and rainfall data. After preprocessing, the air temperature dataset contains the minimum, mean and maximum temperature recorded each day at 87 weather stations in England for each day in 2016 and 2017. The raw air temperature data file is 16.8 MB in size. Similarly, the rainfall dataset contains the total rainfall measured at 69 rainfall stations in England, and is 9.5 MB in size.

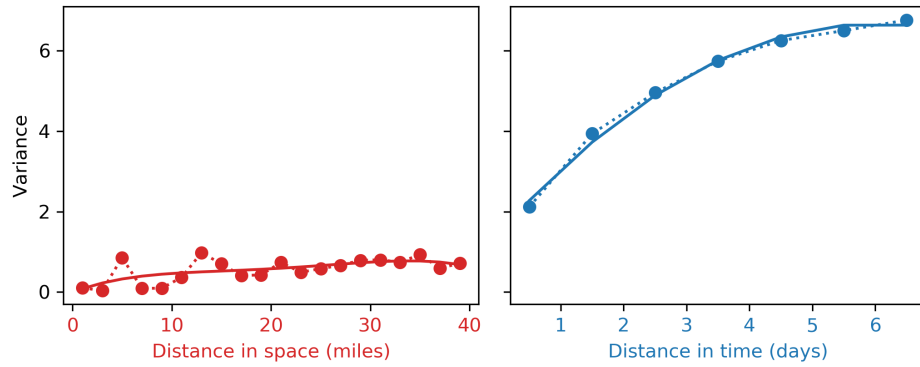
The *mean temperature*, *minimum* and *maximum temperature* features exhibit an approximately normal distribution while the *total precipitation* feature is long-tailed with the most common value recorded for a day being 0 mm rainfall. Figures 5.1b and 5.1c show the experimental variograms of the *mean air temperature* and *daily total precipitation* features respectively. As shown, the variogram value of instances recorded just 1 day apart is quite high for both datasets, yet the variogram value of instances recorded for the same day but up to 30 miles away is much smaller.

### **5.4.2 Linking Procedure and Baseline Augmented Datasets**

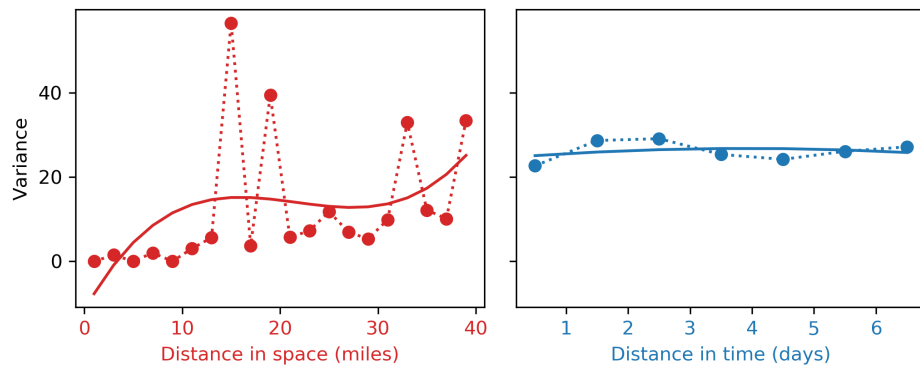
To test the hypotheses outlined in at the start of this section, a set of baseline datasets are created by augmenting the road condition dataset using the raw supplementary datasets and the procedure defined in Algorithm 2. To impute the values of the linked features for each primary instance, the neighbourhood and  $k$  nearest neighbour (kNN) methods are tested, as well as their inverse distance weighted (IDW) variants. A range of parameters are tested for these methods to identify the parameter settings that yield the most accurate imputations (a more detailed discussion is provided in Appendix B). This results in a neighbourhood of 40 miles and 0 days being used to impute the air temperature and rainfall for each primary dataset when using the neighbourhood and IDW neighbourhood methods, and a neighbourhood of 20 miles for the traffic dataset. For the NN and IDW NN techniques, a value of



(a) Daily total traffic count



(b) Daily mean temperature



(c) Daily total precipitation

Figure 5.1: Empirical variograms of features from the supplementary datasets.

$k = 5$  is shown to be most accurate for the air temperature and traffic datasets, while a value of  $k = 1$  is most accurate for the rainfall dataset. Using these values, a baseline dataset for each of the four linking methods is created, each of which is 5.9 MB in size.

After the baseline datasets are created, the supplementary datasets are reduced using  $k$ D-STR with the RES heuristic (Equation 4.8). This provides a baseline of how accurate the linking procedure would be using the naive reduction method that does not consider the primary dataset when reducing the supplementary datasets. Furthermore, the supplementary datasets are reduced using  $k$ D-STR with the LES heuristic (Equation 5.4). When calculating the linked feature error, the LES heuristic used each of the four linking techniques in the error calculation (Equation 5.6). This allows for a direct comparison between the baseline datasets created using the four linking methods, and the augmented datasets created using supplementary datasets that are reduced using the same linking methods. After reducing each of the supplementary datasets using  $k$ D-STR, the primary dataset is augmented using the reduced datasets. For each primary instance, supplementary instances are imputed from each of the reduced datasets at the location of the primary instance for each day between the start and end dates of the primary instance. This allows a comparison to be made between the two supplementary dataset reductions. As well as the 5 values for the parameter  $\alpha$  used in Section 4.3, namely  $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ , the value  $\alpha = 0.01$  is also tested. Since values of  $\alpha$  close to 0 result in more partitions and models being stored (as discussed in Section 4.3), this additional value is tested to measure the impact of an increase in the models stored on the time required to augment the primary dataset.

## 5.5 Results

In this section, a comparison is presented between the accuracy and efficiency of the RES heuristic presented in Chapter 4, and the LES heuristic presented in this chapter. First, a comparison is made of the time taken to augment the primary dataset using the raw and reduced datasets. Second, the error incurred in the linked features is examined, addressing Hypothesis H2.1. Third, the retention of information in space and time is examined, addressing Hypothesis H2.2. Finally, the reconstruction error of the original supplementary datasets is examined, addressing Hypothesis H2.3. Throughout this section, PLR-P modelling is used as a consistent modelling technique across all tests as it provides an average case reduction compared to DCT and DTR modelling (as shown in Figure 4.7), and modelling on partitions is shown to yield a lower NRMSE in many cases. By comparing the results shown in this chapter with the results shown in Section 4.4, results for the other modelling techniques

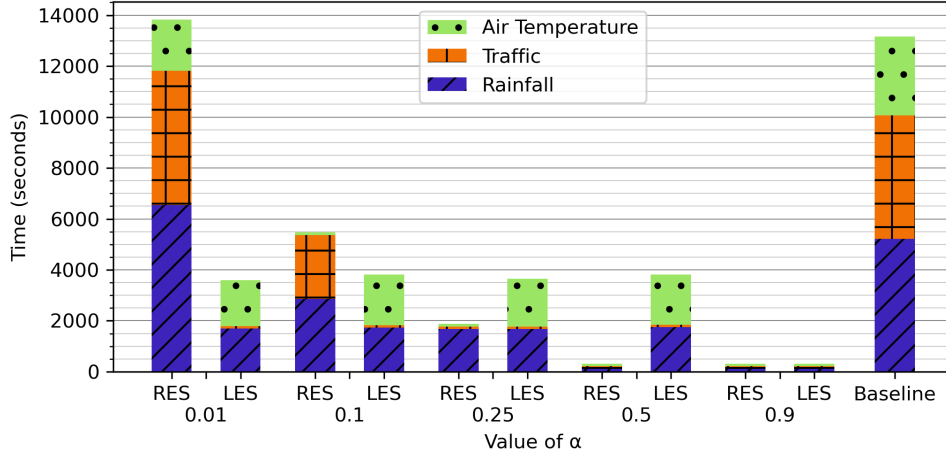


Figure 5.2: Time taken to augment the primary dataset compared to kNN baseline. Results are shown for the RES and LES heuristics.

used in Chapter 4 may be inferred.

### 5.5.1 Speedup Achieved Using Reduced Datasets

Augmenting the primary dataset with the raw supplementary datasets takes between 8,067 and 13,334 seconds using the four linking methods outlined in Section 5.4.2. In comparison, augmenting with the reduced datasets takes between 308 and 329 seconds when  $\alpha = 0.9$ , a value of  $\alpha$  that indicates a strong preference for reducing the quantity of data in the reduced dataset. Conversely, when  $\alpha = 0.01$ , indicating a preference for minimising the error introduced at the cost of increased storage, linking the reduced datasets takes between 3,593 and 3,698 seconds. The time required to link the reduced datasets for increasing values of  $\alpha$ , compared to the baseline augmented dataset created using kNN, is shown in Figure 5.2 (similar results are seen for the other linking methods, as shown in Appendix C.1). These results are for a workstation with an Intel i5-8700k processor and 48GB RAM, utilising a single process per test.

It is important to note that while a maximum speedup of over 13,000 seconds is achieved for augmenting the primary dataset, the time taken to reduce the supplementary datasets can be significantly longer than the time required to link the raw datasets. For example, reducing the supplementary traffic datasets requires between 2,152 and 408,097 seconds. Furthermore, the time required to reduce the air temperature dataset ranges from 269 to 410,807 seconds, and the time required to reduce the rainfall dataset ranges from 7 to 306,017 seconds. Therefore, the speedup achieved by reducing the datasets is typically beneficial when the supplementary datasets are reduced beforehand in an off-line manner. However, as discussed in Chapter 4.4.7, the running time of the  $kD$ -STR algorithm may be improved using a more efficient implementation. This also makes a comparison with the running times of existing methods



difficult.

For the results shown in Figure 5.2, all three reduced datasets contain 1 partition when  $\alpha = 0.9$ . Accordingly, each of the reduced data files use only 2 KB of storage. However, when  $\alpha = 0.01$ , the reduced air temperature dataset contains 1,595 partitions and uses 3.5 MB of storage, and the reduced rainfall dataset contains 2,119 partitions and uses 4.9 MB of storage. These reduced datasets require 1,692 and 1,798 seconds to link respectively, supporting the notion that a reduced dataset containing an increased number of partitions also requires more time to link. The reduced traffic dataset contains 1 partition for all values of  $\alpha$  tested (i.e.  $\alpha = \{0.01, 0.1, 0.25, 0.5, 0.9\}$ ), however the number of coefficients used to store the model of that partition is much larger than the number of coefficients used to store any single partition’s model in the air temperature and rainfall datasets. That the reduced traffic dataset is significantly faster to process than the reduced air temperature and rainfall datasets suggests that the number of partitions has a much greater impact on the time taken to link the reduced datasets, rather than the complexity of the partition models.

In contrast to the results shown for the LES heuristic presented in this chapter, the RES heuristic takes longer to link despite offering no improvement in accuracy. This is a result of the datasets reduced using the RES heuristic storing more partitions, many of which are not used during the augmentation process.

### 5.5.2 Error in Linked Features

By reducing the supplementary datasets, the feature values engineered during linking are different to those engineered using the raw supplementary datasets. For example, Figure 5.3 shows the distribution of feature values for the *mean temperature* and *total precipitation* features. As hypothesised, decreasing the value of  $\alpha$ , which indicates a stronger preference for information retention, increases the number of partitions and models used to store each of the reduced datasets in most cases. When the number of partitions increases, the distribution of the features engineered using the reduced datasets better matches the distribution of features engineered using the raw dataset, either by more accurately matching the range of feature values observed or more closely matching the broad shape of the distribution.

In most cases, decreasing the value of  $\alpha$  lead to a decrease in the maximum error between features engineered using the reduced and raw datasets. As shown in Figure 5.4, the more accurate distribution of engineered values is matched by a more accurate engineered value for each instance when  $\alpha = 0.01$  compared to  $\alpha = 0.9$  (results for all four linking methods can be seen in

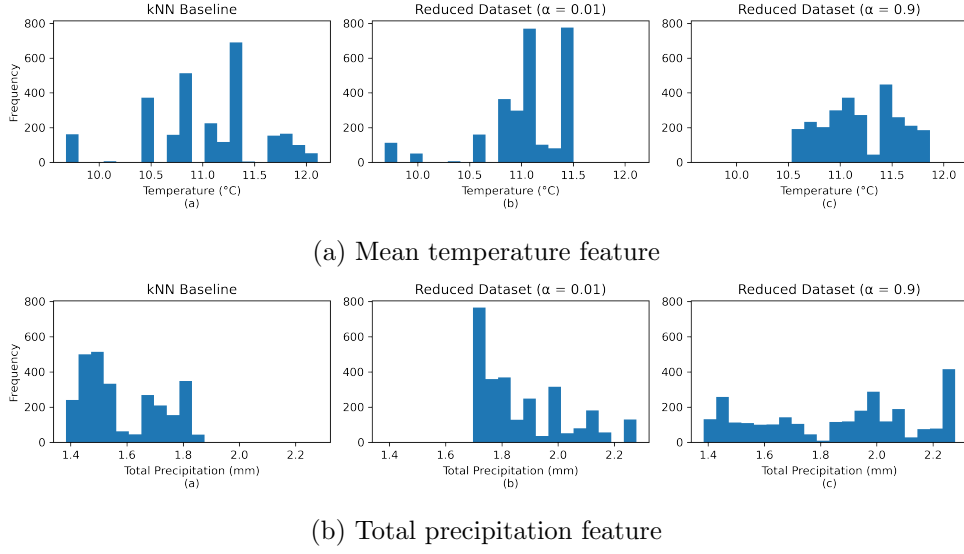


Figure 5.3: Histograms of the *mean temperature* and *total precipitation* features engineered using the kNN baseline method and raw supplementary datasets, and estimating at the road survey location using the reduced dataset, after reduction using the LES heuristic with  $\alpha = 0.01$  and  $\alpha = 0.9$ .

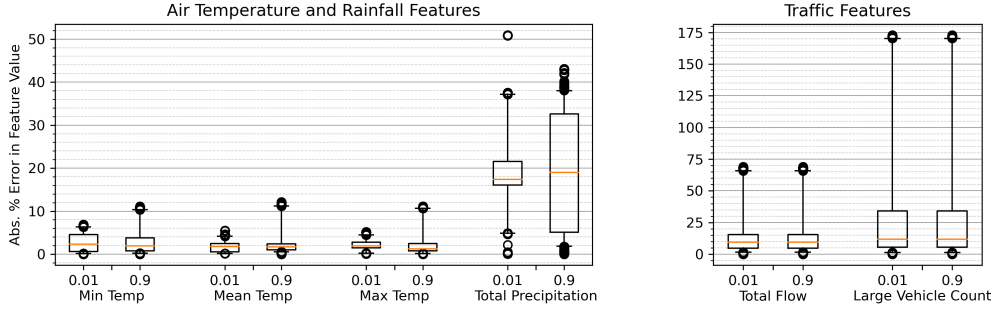


Figure 5.4: Error incurred by engineering features using the reduced datasets (created using the LES heuristic with the kNN linking method). Results are shown for  $\alpha = 0.01$  and  $\alpha = 0.9$ .

Appendix C.2). However, in many cases, such as the *Min Temp* and *Max Temp* features, the median and third quartile errors increase when  $\alpha = 0.01$ . This demonstrates that the engineered feature values become more accurate for some instances as  $\alpha$  decreases, but not all. Note that the higher percentage error of the *total precipitation* feature is caused by the high spatial and temporal variance of the rainfall dataset compared to the lower error and lower spatial variance of the air temperature dataset. For the traffic dataset, the high spatial and temporal variance of the data makes increasing the number of partitions too costly in storage. Thus, when  $\alpha = 0.01$ , the number of partitions stored for the traffic dataset is still 1 and the error incurred is approximately the same as when  $\alpha = 0.9$ .

A comparison of the storage used by each of the three supplementary

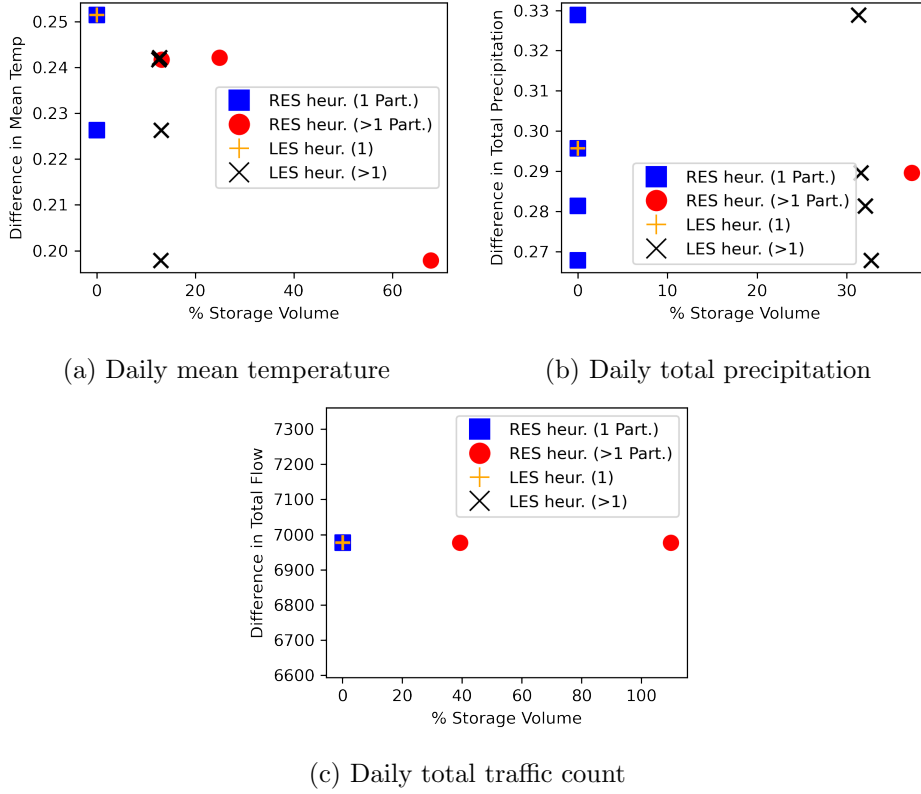
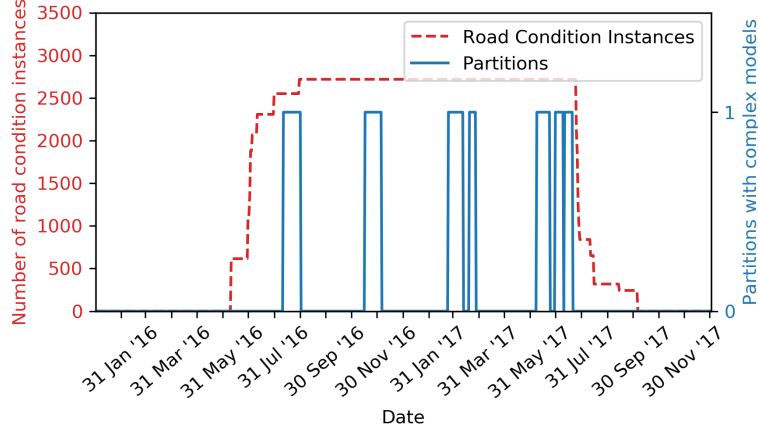


Figure 5.5: Error in engineered features versus storage used by the reduced datasets created using the RES and LES heuristics.

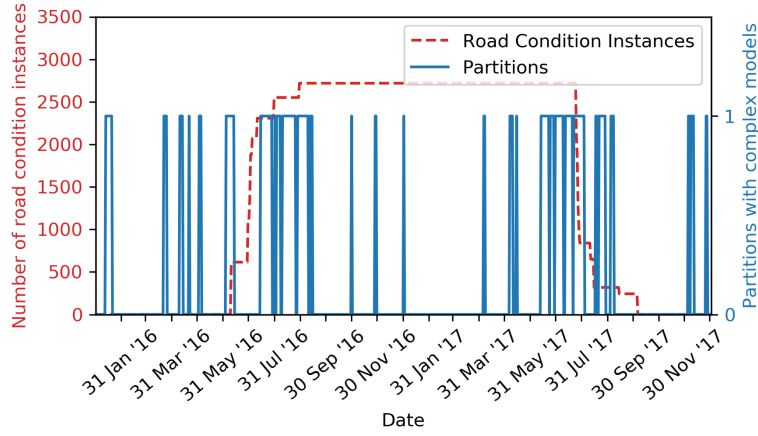
datasets and the error in three engineered features can be seen in Figure 5.5. Each subfigure shows the error versus storage for  $kD$ -STR using the RES and LES heuristics on each of the datasets. The results are split into those reductions that output 1 partition versus those that output more than 1 partition. As expected, the two heuristics are unable to prioritise information retention in the partitions most applicable to the primary dataset when the reduced datasets contain only 1 partition. However, when more than 1 partition is stored, the LES heuristic is able to prioritise information retention in just those partitions that overlap with the areas and time periods that are applicable to the primary dataset. In these results, no reduced dataset that has been reduced using the RES heuristic is able to achieve a lower error and lower storage cost than a dataset reduced using the LES heuristic.

### 5.5.3 Retention of Information in Space and Time

To explore where in space and time  $kD$ -STR used models with a higher number of coefficients (and thus retained more information), the boundaries of the partitions that store more than 1 model coefficient in the reduced datasets are plotted. Figure 5.6 shows the locations of these partitions in time when both heuristics are used to reduce the air temperature dataset and  $\alpha = 0.1$



(a) LES heuristic



(b) RES heuristic

Figure 5.6: Temporal locations of partitions with more than one model coefficient for the air temperature dataset. Results are shown for  $\alpha = 0.1$ .

(results for all datasets and values of  $\alpha$  can be seen in Appendix C.3). As shown, every partition output by  $k$ D-STR with the LES heuristic overlaps with the time period for which instances exist in the primary dataset. That is, only partitions in the reduced dataset that are linked to the primary dataset during linking store more than 1 model coefficient. In contrast, the results for the RES heuristic show that many partitions that are not applicable to the primary dataset store multiple coefficients, meaning  $k$ D-STR chooses to retain information in time periods that are not useful to the primary dataset.

The low variation in the spatial domain of the air temperature dataset results in partitions that cover large spatial areas and short time periods. This means it is likely any partition will overlap with the area that is applicable to the primary dataset. However, the high variation in the temporal domain results in partitions that only cover a small number of time intervals, giving the results shown in Figure 5.6. For the reduced rainfall and traffic datasets, the variance of the datasets in space and time result in partitions with a larger range of spatial areas and time periods. Again, no partition stores more

than 1 model coefficient unless it intersects with the area and time period applicable to the primary dataset when reduced with the LES heuristic, but several partitions that do not intersect store more than 1 model coefficient when reduced with the RES heuristic.

#### 5.5.4 Error in Original Supplementary Features

The error incurred when reconstructing the original supplementary features is higher or approximately equal after reducing with the LES heuristic compared to the RES heuristic, given approximately the same storage volume used. However, this result is only observed when the number of partitions stored for a supplementary dataset is greater than 1, as discussed in Section 5.5.2. Figure 5.7 shows the error incurred for three features when the original supplementary instances are reconstructed from the reduced datasets versus the storage used by the reduced dataset (a comparison for all features can be seen in Appendix C.4). As shown, no reduced dataset that is reduced using the LES heuristic achieves a more accurate reconstruction of the original instances while requiring less storage than the RES heuristic. A more direct comparison for the *mean temperature* feature can be seen in Figure 5.8, where the NRMSE incurred by the RES heuristic is consistently less than or equal to the error incurred by the LES heuristic with the exception of  $\alpha = 0.5$ . This exception is caused by the LES heuristic using more than 1 partition whilst the RES heuristic stores only 1 partition.<sup>5</sup>

### 5.6 Discussion

By reducing supplementary datasets using *kD-STR*, the dataset linking process can be sped up. However, by using the LES heuristic presented in this chapter, a reduction can be achieved that yields similar or higher accuracy of the features engineered in the linking process. More specifically, the key findings were as follows.

First, the LES heuristic leads to a lower maximum error in the engineered features compared to the RES heuristic, when the number of partitions output is greater than 1. The LES heuristic yields a similar or lower NRMSE, given approximately the same storage volume used, and a lower storage volume used given approximately the same NRMSE incurred. This result is seen for all three supplementary datasets, supporting Hypothesis H2.1. Furthermore, when  $\alpha < 0.25$ , indicating a preference for minimising the NRMSE incurred at

---

<sup>5</sup>As the RES and LES heuristics calculate the NRMSE of two different sets of features, namely the original features and linked features, the NRMSE incurred for a given storage ratio may be different for the two heuristics. Thus, a value of  $\alpha$  may yield different NRMSE and storage results for the two heuristics.

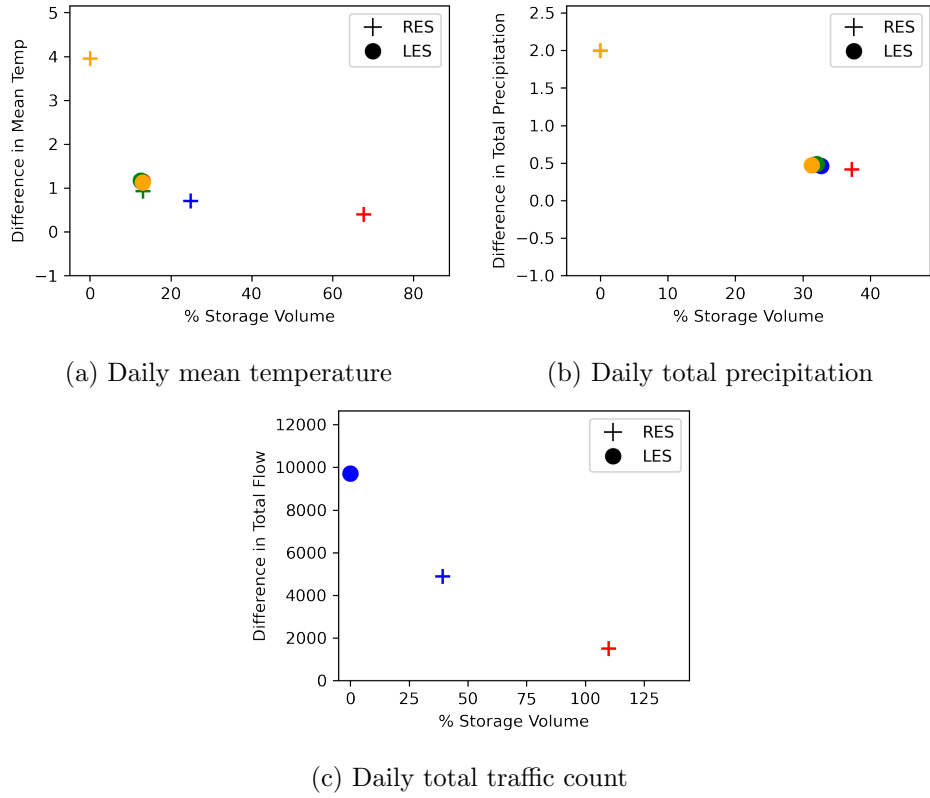


Figure 5.7: Error in original supplementary dataset features versus storage used by the reduced datasets created using the RES and LES heuristics. Only cases where the reduced datasets contained more than 1 partition are shown.

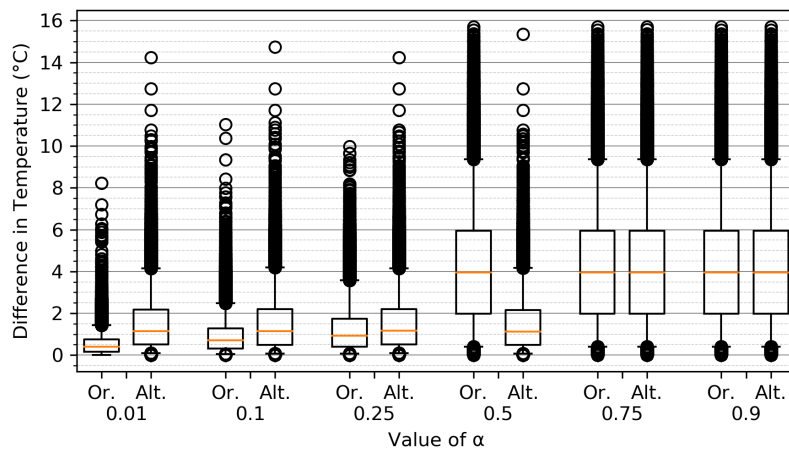


Figure 5.8: Error in mean temperature feature for the original supplementary dataset after being reconstructed from the reduced air temperature dataset. Results are shown for the LES and RES heuristics.

the expense of increased storage overhead, the reduced number of partitions output results in a lower time taken to link the datasets. These two findings demonstrate the improvement achieved by the LES heuristic over the RES heuristic presented in Chapter 4.

Second, the supplementary instances that are far from the primary instances are less accurately reconstructed after being reduced with the LES heuristic compared to the RES heuristic. This demonstrates the efficiency of the LES heuristic in prioritising information retention in just the areas and time periods applicable to the primary dataset, and supports hypothesis H2.3. Though the LES heuristic cannot be used when the primary dataset’s distribution is not known prior to reduction, there are scenarios in which this may be known. For example, datasets such as traffic and air temperature are often collected in monthly or yearly cycles, yet few sensors are removed or added between new samples of the data becoming available. Therefore, the spatial and distributions of the previously collected data can provide adequate and often perfect estimates of the distributions of future data samples. Furthermore, domain knowledge can be used to supplement the previously collected data, allowing the LES heuristic to be used.

Third, no partition in the reduced dataset that does not overlap with the primary instances stores more than a single model coefficient when the LES heuristic is used. In comparison, multiple partitions that do not overlap with the area applicable to the primary dataset retain more than 1 model coefficient when the RES heuristic is used. These coefficients retain information that is not useful for the linking process, giving a less efficient reduction of the supplementary datasets. Furthermore, these results support hypothesis H2.2 by demonstrating the efficiency improvement offered by the LES heuristic. However, the variance of the instances within a partition is also affected the number of model coefficients stored for that partition. Therefore, future work may investigate the extent to which the number of coefficients stored for a partition is attributable to the magnitude of accuracy improvement in the linked features versus the number of primary instances that the model is linked to.

## 5.7 Summary

In this chapter, a wrapper method for integrating  $k$ D-STR data reduction with the data linking process has been presented. This integrated process allows the input datasets to be reduced, thereby speeding up the linking process, and allows the linked dataset to be reduced for faster analysis. Furthermore, an alternative heuristic for the  $k$ D-STR algorithm, namely the LES heuristic, has been presented for reducing supplementary datasets prior to reduction.

While the RES heuristic presented in Chapter 4 is shown to be effective at reducing spatio-temporal datasets to smaller representations while minimising the NRMSE incurred, it does not consider the spatial and temporal distributions of the primary dataset when reducing supplementary datasets. The LES heuristic presented in this chapter is shown to give a more efficient reduction of the supplementary datasets in linking scenarios, while achieving a lower or similar NRMSE as the RES heuristic. Unlike the RES heuristic, the LES heuristic requires background knowledge on the appropriate linking methods for the datasets being reduced. However, as the heuristic specifically targets the data linking scenario, this domain knowledge is assumed to have been captured in an offline style prior to reduction.



## Chapter 6

# Implications of Real-World Applications on Spatio-Temporal Reduction

In Chapter 4, the  $k$ D-STR algorithm was introduced for reducing large spatio-temporal datasets, thereby reducing the memory and time required to store and process the data. However, as the quantity of data in a dataset continues to increase, the properties of ‘big data’ may place practical limitations on how and when  $k$ D-STR may be used. Specifically, three qualities of real-world large datasets may limit the utility of the reduction process in an offline setting, namely, *volume*, *heterogeneity* and *veracity* [108]. These qualities limit how much data can be reduced by  $k$ D-STR, the reduction in storage achieved, and the error incurred.

The first characteristic, volume, affects the time taken to reduce the dataset. Though the aim of data reduction is to reduce the volume of a dataset, the computational complexity of the algorithm limits the quantity of data that can be reduced. Furthermore, when a dataset contains too many features or instances, it must be reduced in a distributed manner wherein the data is spread across multiple *compute nodes* and reduced by the compute nodes collectively [86]. By reducing the data in a distributed computing environment, the data may be reduced to a smaller form that can be processed more efficiently.

However, it is ambiguous how the partitioning process of the  $k$ D-STR algorithm should perform in a distributed environment. Though distributed clustering algorithms have been proposed that can be used by  $k$ D-STR, the partitions generated by the clustering step may span multiple machines. Thus, the partitioning process incurs a significant communication overhead which can make the reduction process infeasible. Furthermore, the communication overhead incurred when modelling partitions that span multiple machines may be too great. Therefore, this chapter addresses these issues and proposes an

adapted algorithm — Distributed  $k$ D-STR, referred to as D $k$ D-STR.

The second limiting characteristic, heterogeneity, refers to the analysis and processing of multiple datasets simultaneously. In Chapter 5, the  $k$ D-STR algorithm was integrated with the data augmentation process and an alternative heuristic presented for reducing supplementary datasets. Yet, in other contexts, multiple datasets may be stored and processed together that do not have an inherent primary and supplementary relationship. For example, in co-occurrence mining, datasets are processed together by merging the spatio-temporal domains of the datasets into a single spatio-temporal domain. When processing reduced datasets in such scenarios, each partition in one reduced dataset may be linked to multiple partitions in each of the other datasets. This complicates the discovery of co-occurring events and trends, and increases the overhead of linking the reduced datasets significantly. Thus, for such scenarios it would be beneficial to partition the datasets into a single set of partitions in the spatio-temporal domain. That is, the partition boundaries should be determined with reference to the instances in all of the datasets. By reducing the datasets collectively to a single set of partitions, the overhead of linking the reduced datasets is minimised, allowing the datasets to be linked in their reduced form. This chapter therefore introduces the Mutual  $k$ D-STR algorithm (M $k$ D-STR), an adaptation of  $k$ D-STR for reducing multiple datasets simultaneously.

The third limiting characteristic, veracity, refers to the impact of noise, error and missing data on the reduced dataset [104]. In many domains, datasets are affected by noise, error and missing data. Though preprocessing tasks may be used to clean the data, these processes may be imperfect and may not be appropriate for a given scenario. For example, outlier removal may remove instances that accurately capture disruptions in the spatio-temporal process, such as road traffic accidents in the traffic dataset. In such cases, the dataset must be reduced and stored in its unclean form. Therefore, it is important to understand how the presence of noise, error and missing data may affect the reduction in storage achieved and error incurred. This chapter therefore investigates the impact of veracity on  $k$ D-STR.

Thus, this chapter makes three contributions. First, Section 6.1 presents Mutual  $k$ D-STR, an adaptation of  $k$ D-STR for mutually reducing multiple datasets simultaneously. This adaptation uses a concept found in partitioning literature, yet overcomes a weakness inherent in the existing method. Second, Section 6.2 presents Distributed  $k$ D-STR, an algorithm which overcomes ambiguities on how to perform data reduction in a distributed manner. Furthermore, empirical analysis is used to demonstrate the speedup achievable when reducing data in a distributed environment, thereby allowing larger datasets to be reduced. Third, Section 6.3 assesses the utility of  $k$ D-STR when reducing

datasets containing noise, missing data and location and time error. Through empirical analysis, the effect of these issues on reduced datasets is investigated. These results indicate the reduction that may be achieved for other datasets that exhibit similar distributions and characteristics to the perturbed datasets used.

## 6.1 Mutual Reduction

In Chapter 4, the  $k$ D-STR algorithm was presented for reducing a single spatio-temporal dataset and, in Chapter 5, an alternative heuristic was presented for reducing a supplementary dataset in the context of a primary dataset. However, in many applications, multiple datasets that cover the same area and time period may be stored and processed together. The datasets may not have a primary and supplementary relationship, yet we may wish to analyse them simultaneously. For example, in co-occurrence mining, datasets are decomposed into sets of partitions, where the spatial and temporal bounds of the partitions are the same across each of the datasets. Techniques such as the anomaly detection method presented by Zhang *et al.* place instances from multiple datasets into a single user-defined partitioning scheme to detect co-occurring events [145]. In contrast, the partitioning technique used by  $k$ D-STR uses the feature values of a dataset to partition the data in the spatio-temporal domain. Thus, each partition in one dataset may overlap with multiple partitions in each of the other datasets. This makes the discovery of co-occurring events difficult, as the events in one partition may correlate with the events in multiple other partitions. Furthermore, the overhead of linking the partitions is increased. Therefore, it would be beneficial to mutually partition the datasets, with the instances in all of the datasets used to create a single partitioning in the spatio-temporal domain.

However, it is ambiguous how the  $k$ D-STR algorithm should be adapted for mutually reducing multiple datasets. Since the datasets may contain features that are incomparable, the instances from different datasets cannot be compared in the feature space. In the databases community, this issue has been considered by the AdaptDB partitioning technique [85]. AdaptDB aims to speed up the retrieval of instances from multiple datasets that are linked by one or more common features. AdaptDB recursively partitions each of the datasets independently for a single feature that must be specified before partitioning. The partitioning continues until each partition can fit into the memory of a single compute node, although the partitions are split on their median values rather than the similarity of their instances. In the context of spatio-temporal data, the features used to partition the instances are the spatial and temporal referencing features, however the user must select one of the spatio-temporal

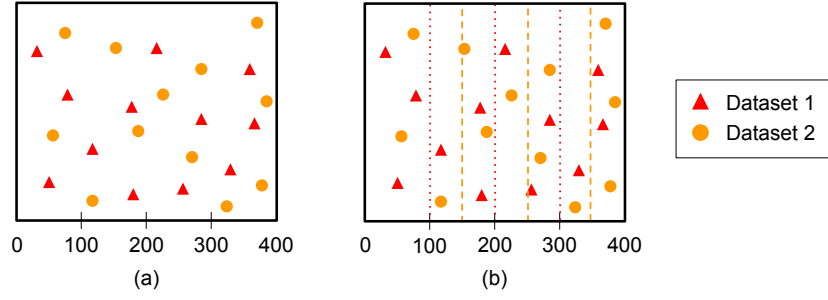


Figure 6.1: Initial partitioning of two datasets by AdaptDB [85]. In (a), the instances from datasets are shown in the common 2D space. In (b), the two datasets have been independently partitioned into equal-sized partitions, yielding two independent partitionings of the 2D space.

features to partition prior to reduction. If the spatio-temporal distribution of instances is different for each of the datasets, the partition boundaries will differ between each of the datasets. To fix this issue, AdaptDB iteratively moves blocks of instances between partitions within each dataset as the data is retrieved, thereby minimising the number of partitions in one dataset that overlap with each partition in the other datasets.

Consider the example shown in Figure 6.1. In subfigure (a), the instances are shown in the 2-dimensional spatial domain. In subfigure (b), the two datasets have been partitioned along the  $x$  axis separately to give 4 equally sized partitions per dataset. For dataset 1, the partition boundaries are  $[0, 100)$ ,  $[100, 200)$ ,  $[200, 300)$ ,  $[300, 400)$ . For dataset 2, the partition boundaries are  $[0, 150)$ ,  $[150, 250)$ ,  $[250, 350)$ ,  $[350, 400)$ . After this initial partitioning, AdaptDB allows users to query the database and retrieve instances by providing a desired range of the  $x$  axis. As the queries are processed, AdaptDB monitors each partition and counts how many other partitions are retrieved alongside that partition. If moving subsets of instances from one partition to another reduces the overall number of overlapping partitions, that change is made. Furthermore, if other features are used to link the datasets within user queries, such as the  $y$  axis, the partition tree adapts to accommodate the other features. Therefore, the boundaries of the partitions for each dataset adapt over time and may converge to a set of partition bounds that are the same for all of the datasets.

However, this adaptive approach may yield quite different partition schemes initially and take a long time to converge. While this may not impair AdaptDB’s aim of speeding up query answering, it cannot guarantee the set of partitions are the same across all of the datasets. Furthermore, only one feature is used to partition the data initially, and the partitioning technique does not consider the similarity of the instances within each dataset. Therefore, such an approach is not appropriate for  $k$ D-STR, although the procedure of partitioning

each dataset independently then merging these partitioning schemes may be advantageous. An adapted  $k$ D-STR algorithm is therefore required that can create a single set of partitions in the spatio-temporal domain for all of the datasets, and use the similarity of the instances to inform the partitioning scheme. The adapted algorithm must output a model per partition for each dataset, so that instances can be retrieved or imputed from any partition for each dataset. Such an adapted method is presented in Section 6.1.1, and is referred to as the Mutual  $k$ D-STR algorithm (MkD-STR). This adapted algorithm builds upon the approach taken by AdaptDB but does not require iterations of query answering to find a common partitioning of the datasets.

### 6.1.1 Mutual Partitioning Technique for $k$ D-STR

Given a set of  $n$  datasets,  $D^1, \dots, D^n$ , MkD-STR aims to reduce the datasets to a single set of partitions  $P$  in the  $S \times T$  space, and  $n$  sets of models. Similar to the aim of partitioning in Chapter 4, every instance in  $D^1, \dots, D^n$  must belong to exactly one partition  $\forall p_i, p_j \in P : p_i \cap p_j = \emptyset$  and  $\cup_{i=1}^{|P|} D_{p_i} = \cup_{l=1}^n D^l$ . When partitioning the  $S \times T$  space, MkD-STR makes use of the feature values and spatio-temporal locations of every instance in each dataset. To overcome the incomparability of the datasets' features, an approach inspired by both AdaptDB and the partitioning technique presented in Chapter 4 is used. Each dataset is first partitioned independently using the feature values of the instances to form  $n$  sets of partitions,  $P^l$  where  $1 \leq l \leq n$ . These sets of partitions are then combined to form a single set of partitions  $P$ . However, while AdaptDB iteratively moves partition boundaries and may converge to a mutual partitioning of all of the datasets, this can take time and convergence is not guaranteed. To overcome this, the partitions of the datasets can be *intersected*. That is, for every combination of partitions from all of the datasets, the intersection of their spatial and temporal bounds is used to define a new partition in  $P$ :

$$P = \{p_{\text{int}} \mid \forall p_i \in P^k, \forall p_j \in P^l : p_{\text{int}} = \text{intersect}(p_i, p_j)\} \quad (6.1)$$

where  $\text{intersect}(p_i, p_j)$  gives the intersection of partitions  $p_i$  and  $p_j$ ,  $1 \leq k, l \leq n$ , and  $k \neq l$ . This process can be seen in the simple example in Figure 6.2. In (a), the two datasets are first partitioned independently using the process presented in Chapter 4. In (b), these two partitionings of the spatio-temporal domain are combined to form a single partitioning. For example, the partition  $p_5$  is the result of intersecting  $p_1$  of Dataset 1 with partition  $p_3$  of Dataset 2. This process results in a single set of partitions, where each resulting partition  $p_i \in P$  is linked to exactly one partition from each of the constituent datasets. Thus, for each dataset  $D^l$ , a model can be formed per partition in  $P^l$ , and

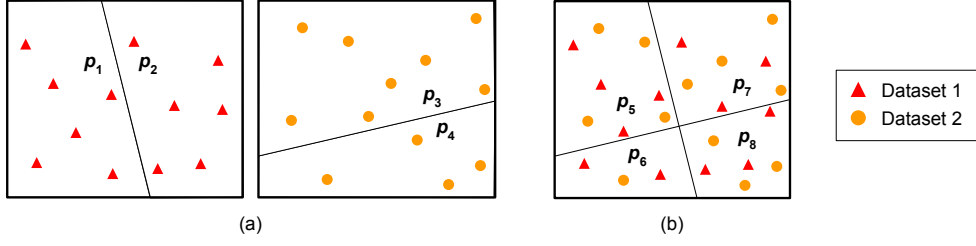


Figure 6.2: Partitioning of two datasets by MkD-STR. In (a), the two datasets are partitioned independently. In (b), the two partitionings have been combined to yield 4 partitions of the spatio-temporal domain.

each partition in  $P$  linked to the appropriate model for each dataset.

To reduce multiple datasets using the  $k$ D-STR algorithm, several adaptations have to be made. First, the number of clusters chosen for each dataset should be independent of the number chosen for each of the other datasets. Though the same number may be chosen for each, this may result in a less efficient reduction as more partitions and models may be stored for some datasets unnecessarily. Thus, the *clusterTree* and *numberClusters* variables in Algorithm 1 (Section 4.2.3) are replaced with  $n$  cluster trees and an  $n$ -dimensional vector of cluster identifiers, where the number of clusters chosen for  $D^l$  is set in the  $l$ th value of the vector. Second, an additional step of partitioning is performed that combines the set of partitions for each dataset into a *global* set of partitions  $P$ . These partitions are inherently linked to a partition from each dataset, and so are also linked to exactly one model  $m^l$  per dataset  $D^l$ . Finally, the iterative process of the reduction algorithm is adapted to test the reduction in heuristic value achieved when increasing the number of partitions for one of the datasets, or increasing the model complexity of one of the dataset models. Each dataset is tested to see if increasing its number of partitions improves the heuristic value, and is tested again to see if increasing the number of coefficients stored for one of its models improves the heuristic value further.

An overview of the MkD-STR algorithm for mutually reducing multiple datasets can be seen in Figure 6.3. In (a), the instances of the two input datasets are shown in the 2-dimensional spatio-temporal domain independently<sup>1</sup>. First, MkD-STR clusters and partitions the two datasets independently to form a partition tree per dataset. Then, as shown in (b), MkD-STR begins at the root of each partition tree and forms a single partition in the spatio-temporal domain by intersecting the partitions from the root of each partition tree. This partition includes all instances from both datasets, and a model of the simplest form is formed of this partition for each dataset. After the model is fitted to the data, the result of the heuristic  $heuristic(\langle D^1, \dots, D^n \rangle, P, \langle M^1, \dots, M^n \rangle)$  is calculated for the first reduction step.

<sup>1</sup>Similar to  $k$ D-STR, MkD-STR may be used for any number of spatio-temporal dimensions.

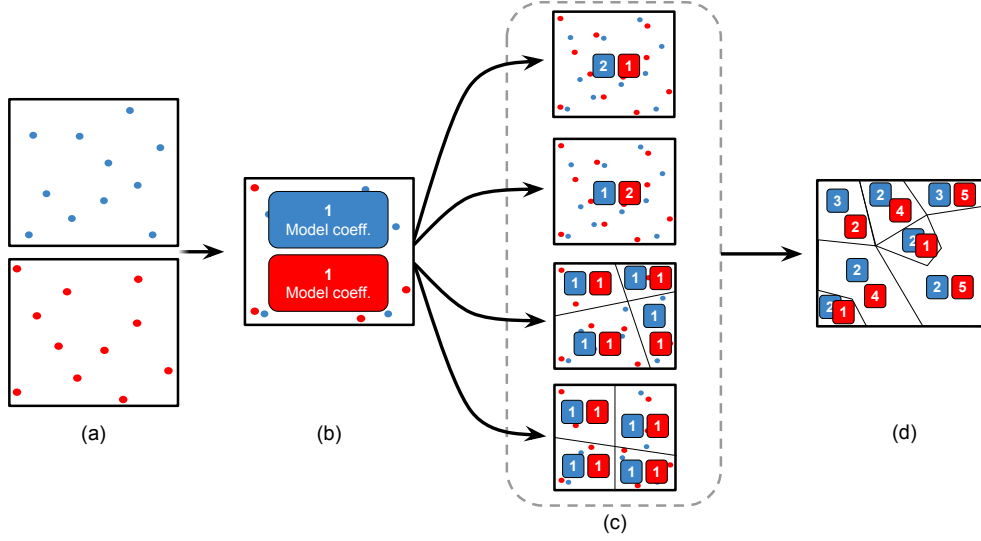


Figure 6.3: Overview of MkD-STR data reduction. (a) The input datasets. (b) Spatial and temporal domain partitioned into 1 partition, and 1 model coefficient stored per dataset. (c) Number of model coefficients and number of partitions repeatedly increased and tested until the objective function is minimised. (d) Partitions and model coefficients output.

After the first reduction step, the algorithm iterates as shown in (c). On each iteration, MkD-STR decides whether to increase the complexity of one of the existing models, or increase the number of partitions in the partitioning of the  $\mathbb{S} \times \mathbb{T}$  space. To increase the number of partitions, MkD-STR can iterate one level down in either partitioning tree and intersect the partitions of this tree with the partitions on the current level for each of the other datasets' partitioning trees. Thus, for 2 datasets, MkD-STR may increase the number of coefficients stored for an existing model in either of the datasets, or increase the number of partitions by increasing the number of constituent partitions for either of the datasets. On each iteration the move that minimises the heuristic value is taken. When the heuristic value cannot be minimised further, MkD-STR terminates and outputs the set of partitions and set of models  $P, \langle M^1, \dots, M^n \rangle$ , as shown in (d).

The MkD-STR algorithm is shown in Algorithm 4. Here, lines 1—11 are an adaptation of lines 1—8 of Algorithm 1 (Section 4.2.3). Line 1 establishes the  $n$ -dimensional vector of the number of clusters used to partition each dataset, with  $nClusters^l$  used to refer to position  $l$  of the vector. Lines 2—8 form a partition tree, single partition and corresponding model for each dataset, with the  $i$ th partition and model of dataset  $D^l$  denoted  $p_i^l$  and  $m_i^l$  respectively. Lines 9 and 10 form the global set of partitions and calculate the heuristic value for storing 1 global partition and 1 model per dataset. After this initialisation step, lines 11—23 iteratively test the effect of increasing the number of partitions stored and the complexity of each model until the

heuristic value cannot be minimised further. These lines correspond to lines 9—36 of Algorithm 1, although for ease of presentation the subsections of the algorithm have been separated into the functions *incrCoefficients()* and *incrPartitions()* (Algorithms 5 and 6 respectively). The value  $l_{\text{best}}$  is used to denote which dataset minimised the values of  $h_1$  and  $h_2$ . After the MkD-STR algorithm finishes iterating, the set of global partitions  $P$  is output, along with  $n$  sets of models, where each partition in  $P$  is linked to one model in each of the  $n$  sets.

---

**Algorithm 4:** Mutual  $k$ D-STR algorithm for reducing multiple data-sets simultaneously

---

**Data:** Input datasets  $D^1, \dots, D^n$   
**Result:** Set of partitions and models  $P, \langle M^1, \dots, M^n \rangle$

```

1 nClusters =  $\langle 1, \dots, 1 \rangle$  //  $n$ -d vector of partition tree levels
  // First, partition each dataset and create set of global partitions
2 for  $1 \leq l \leq n$  do
3   cTreel = cluster( $D^l$ )
4    $P^l \leftarrow \text{findPartitions}(D^l, \text{cTree}^l, \text{nClusters}^l)$ 
5    $M^l \leftarrow \emptyset$  // Initialise models for  $D^l$  to the empty set
6   for  $p_i^l$  in  $P^l$  do
7      $m_i^l \leftarrow \text{model}(D_{p_i}^l, 1)$  // Model  $p_i^l$  using the simplest complexity
8      $M^l.\text{add}(m_i^l)$ 
9  $P \leftarrow \text{intersection}(P^1, \dots, P^n)$  // Create global partitions
10  $h \leftarrow \text{heuristic}(\langle D^1, \dots, D^n \rangle, P, \langle M^1, \dots, M^n \rangle)$  // 1P, 1M per  $D^l$ 
  // Second, iteratively increase model complexities and partitions
11 do
12    $h_1, M_{\text{best}}, l_{\text{best}} \leftarrow \text{incrCoefficients}(\langle D^1, \dots, D^n \rangle, P, \langle M^1, \dots, M^n \rangle)$ 
13    $h_2, l_{\text{best}} \leftarrow \text{incrPartitions}(\langle D^1, \dots, D^n \rangle, P, \langle M^1, \dots, M^n \rangle, \text{cTrees}, \text{nClusters})$ 
14   if  $h_1 < h_2$  and  $h_1 < h$  then
15      $M^{l_{\text{best}}} \leftarrow M_{\text{best}}$ 
16      $h \leftarrow h_1$ 
17   else if  $h_2 < h_1$  and  $h_2 < h$  then
18      $P \leftarrow P'$ 
19      $P^{l_{\text{best}}} \leftarrow P^{l_{\text{best}}'}$ 
20      $M^{l_{\text{best}}} \leftarrow M^{l_{\text{best}}''}$ 
21      $h \leftarrow h_2$ 
22      $\text{nClusters}^{l_{\text{best}}} = \text{nClusters}^{l_{\text{best}}} + 1$ 
23 while  $h_1 < h$  or  $h_2 < h$ 
24 return  $P, \langle M^1, \dots, M^n \rangle$ 

```

---



---

**Algorithm 5:** *incrCoefficients()* function for MkD-STR Algorithm  
(Algorithm 4)

---

**Data:** Input datasets  $D^1, \dots, D^n$ , global partitions  $P$ , vector of sets of models  $\langle M^1, \dots, M^n \rangle$

**Result:** Heuristic value  $h_1$ , set of models  $M_{\text{best}}$ , dataset yielding lowest heuristic value  $l_{\text{best}}$

```

12  $h_1 \leftarrow \infty$ ;
13 for  $1 \leq l \leq n$  do
14     for  $m_i^l$  in  $M^l$  do
15          $M'' \leftarrow M^l$ ;
16          $m_i'' \leftarrow \text{model}(D_{p_i}^l, m_i^l.\text{complexity} + 1)$ ;
17          $M''.\text{replace}(m_i^l, m_i'')$ ; // Replace  $m_i^l$  with  $m_i''$  in  $M''$ 
18          $h' \leftarrow \text{heuristic}(\langle D^1, \dots, D^n \rangle, P, \langle M^1, \dots, M^n \rangle)$ ; // Calculate
            heuristic with  $M''$ 
19         if  $h' < h_1$  then
20              $h_1 \leftarrow h'$ ;
21              $M_{\text{best}} \leftarrow M''$ ;
22              $l_{\text{best}} \leftarrow l$ ;
23 return  $h_1, M_{\text{best}}, l_{\text{best}}$ 

```

---

**Algorithm 6:** *incrPartitions()* function for MkD-STR Algorithm  
(Algorithm 4)

---

**Data:** Input datasets  $D^1, \dots, D^n$ , global partitions  $P$ , vector of sets of models  $\langle M^1, \dots, M^n \rangle$ , vector of cluster trees  $cTrees$ , vector of partition tree levels  $nClusters$

**Result:** Heuristic value  $h_1$ , set of models  $M_{\text{best}}$ , dataset yielding lowest heuristic value  $l_{\text{best}}$

```

1  $h_2 \leftarrow \infty$ ;
2 for  $1 \leq l \leq n$  do
3      $P'' \leftarrow \text{findPartitions}(D^l, cTree^l, nClusters^l + 1)$ ;
4      $M''' \leftarrow \emptyset$ ;
5     for  $p_i^l$  in  $P''$  do
6         if  $p_i^l$  in  $P^l$  then
7              $M'''.\text{add}(m_i^l)$ ; // Add  $m_i^l \in M^l$  to  $M'''$ 
8         else
9              $m_i''' \leftarrow \text{model}(D_{p_i}^l, 1)$ ;
10             $M'''.\text{add}(m_i''')$ ;
11     $P' \leftarrow \text{intersection}(P^1, \dots, P^n)$ ; // Create global partitions with
         $P''$ 
12     $h'' \leftarrow \text{heuristic}(\langle D^1, \dots, D^n \rangle, P', \langle M^1, \dots, M^n \rangle)$ ; // Calculate
        heuristic with  $P'$  and  $M'''$ 
13    if  $h'' < h_2$  then
14         $h_2 \leftarrow h''$ ;
15         $l_{\text{best}} \leftarrow l$ ;
16 return  $h_2, l_{\text{best}}$ 

```

---

### 6.1.2 Experimental Evaluation Methodology

To evaluate the impact of mutually reducing multiple datasets together, the datasets presented in Chapter 3 are reduced in combinations of 2 and 3 datasets. In Section 6.1.3, the results of these comparisons are presented and compared against the results presented in Chapter 4. Here, the term *independently reduced datasets* is used to refer to the datasets after they are reduced separately from each other (i.e. with  $k$ D-STR), and the term *mutually reduced datasets* is used to refer to the datasets after they are reduced together (i.e. with  $Mk$ D-STR). For both methods, PLR-P modelling is used as a consistent modelling technique across all tests as it provides an average case reduction compared to DCT and DTR modelling (as shown in Figure 4.7), and modelling on partitions is shown to yield a lower NRMSE in many cases. By comparing the results shown in this Chapter with the results shown in Section 4.4, results for the other modelling techniques used in Chapter 4 may be inferred. Furthermore, the same 5 values of alpha used in Chapter 4 are tested —  $\alpha \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ . Specifically, 2 hypotheses are tested:

H3.1 *In some cases, more partitions are output for the mutually reduced datasets than the independently reduced datasets, although in cases the opposite is true.*

That is, the intersection of partitions may yield multiple new partitions, therefore the total number of partitions created by the mutual reduction may be higher. However, when this is true, the storage overhead of the reduced datasets is increased, and this may prevent  $Mk$ D-STR from iterating as far down the partition tree as  $k$ D-STR. Consequently, the number of partitions output for some datasets will be lower when the datasets are reduced mutually, and the total number of partitions may be lower than the total number of partitions output when the datasets are reduced independently.

H3.2 *The NRMSE of a dataset may not decrease as the number of partitions output increases.*

When two datasets are mutually reduced, the intersection of their partitions may result in many more partitions being output. However, each of these resultant partitions will be linked back to a single partition for a given input dataset, and so multiple partitions are linked to the same model for each dataset. Thus, although the number of partitions output by  $Mk$ D-STR may be higher than the number output by  $k$ D-STR for a dataset, the number of models output may be lower. As a result, the NRMSE of some reduced features may not decrease as the number of partitions output by  $Mk$ D-STR increases. This hypothesis contradicts the

results shown in Chapter 4, in which increasing the number of partitions in the reduced dataset decreases the NRMSE incurred.

### 6.1.3 Results

The results of mutually reducing the datasets presented in Chapter 3 are discussed in this section. For each of the reduced datasets, the results of MkD-STR are contrasted with the results achieved when reducing the datasets independently using  $k$ D-STR. For each dataset, the term *mutually reduced* is used to refer to the dataset after it has been reduced mutually with one or more other datasets using MkD-STR, and the term *independently reduced* is used to refer to the dataset after it has been reduced by itself using  $k$ D-STR.

#### Effect on Partitioning

Similar to the results presented in Chapter 4, the number of partitions output during the mutual reduction process is correlated with the spatio-temporal variability of the input datasets. However, when mutually reducing the datasets, the higher variability of the traffic dataset dominates the resultant spatio-temporal partitioning compared to the air temperature and rainfall datasets. That is, more partitions are generated for the traffic dataset than the air temperature and rainfall datasets prior to being intersected. As a result, more models are output for the traffic dataset than the air temperature and rainfall datasets, with the air temperature dataset also dominating the rainfall dataset. The number of partitions generated for each dataset by MkD-STR prior to being intersected is shown in Figure 6.4, as well as the number of partitions generated by  $k$ D-STR for comparison. The total number of partitions output for each combination of datasets is denoted *Mutual (all)*. Note that the number of partitions generated for each dataset by MkD-STR prior to being intersected is equal to the number of models output by MkD-STR for that dataset.

In all cases, the number of models generated by the mutual reduction process for the air temperature and rainfall datasets is lower than or equal to the number output when the datasets are reduced independently. In most cases, the mutually reduced traffic dataset contains more than or equal to the number of models when the dataset is reduced independently. For example, when all three datasets are mutually reduced and  $\alpha = 0.25$ , the mutually reduced output contains 6,013, 1 and 2 models for the traffic, air temperature and rainfall datasets respectively compared to 5,182, 637 and 3 models when the datasets are reduced independently. The only exceptions to this occur when  $\alpha = 0.1$  and the traffic dataset is mutually reduced with the air temperature dataset, and when all three datasets are mutually reduced. Thus, a higher number of partitions may be output for highly varied datasets when mutually reduced

with other datasets, although fewer partitions may be output for datasets with low spatio-temporal variance.

When  $\alpha = 0.1$ , mutually reducing the datasets results in the reduced dataset containing fewer partitions than the sum of partitions output when the datasets are reduced independently. For example, when all three datasets are mutually reduced and  $\alpha = 0.1$ , the mean sum of partitions output is 19,830 while the mutually reduced datasets contain 6,333 partitions on average. Consequently, the mutually reduced datasets also use less storage than the total storage used when the same datasets are reduced independently. When  $\alpha \geq 0.1$ , mutually reducing the datasets results in more or an equal number of partitions output than the sum of partitions output when the datasets are reduced independently. The only exception is when  $\alpha = 0.25$  and the air temperature and rainfall datasets are mutually reduced. When  $\alpha = 0.9$  and in most cases when  $\alpha = 0.75$ , the number of partitions output by both the independent and mutual reduction processes is 1. Therefore, mutually reducing the datasets only affects the partitioning of the data when  $\alpha < 0.9$ .

### **Effect on Error and Storage Used**

The quantity of storage used by the mutually reduced datasets is directly proportional to the number of partitions in the mutually reduced dataset (as shown in Appendix D.1). Accordingly, when the number of partitions in the mutually reduced datasets is less than the total number of partitions in the independently reduced datasets, the storage used by the mutually reduced dataset is less than the total storage used by the independently reduced datasets. Similarly, when the number of partitions in the mutually reduced datasets is higher, the storage used is also higher than the total storage used by the independently reduced datasets. For example, when all three datasets are mutually reduced and  $\alpha = 0.1$ , the mean total storage used by the independently reduced datasets is 16.4% of the original storage volume while the mutually reduced dataset uses 5.0%. The storage used by the independently and mutually reduced datasets can be seen in Table 6.1.

When  $\alpha = 0.1$ , the number of partitions and models in the mutually reduced dataset is lower than the independently reduced datasets. As a result of fewer models being output, the mean NRMSE incurred is higher than or equal to the NRMSE incurred by the independent reductions in all cases. This can be seen in Figure 6.6, which shows the NRMSE incurred when reducing the traffic and air temperature datasets, traffic and rainfall datasets, and all 3 datasets together (a legend for results these results is shown in Figure 6.5, and results for all four combinations of datasets can be found in Appendix D.2). When all three datasets are mutually reduced and  $\alpha = 0.1$ , the NRMSE incurred

Table 6.1: Storage used by the independently and mutually reduced datasets. A value of  $\alpha = 0.1$  indicates high storage used for minimised error, and  $\alpha = 0.9$  indicates minimised storage but higher error.

Dataset(s)	$\alpha = 0.1$		$\alpha = 0.9$	
	Min (MB)	Max (MB)	Min (MB)	Max (MB)
Air Temperature	0.1	25.8	0.1	5.2
Traffic	0.7	69.3	0.1	32.3
Rainfall	0.1	11.9	0.1	3.1
Air Temp. & Rain.	0.2	13.0	0.1	0.1
Traffic & Air Temp.	1.6	19.8	0.1	0.1
Traffic & Rain.	1.6	27.7	0.1	0.1
All 3 datasets	1.7	10.9	0.1	0.1

for the *total volume* feature is 5.9% and the *temperature* feature is 12.5%. In comparison, the mean NRMSE incurred by the independently reduced datasets is 4.7% and 5.2% respectively. For the air temperature and rainfall datasets, the reduced number of models output by the mutual reduction process leads to a higher or equal NRMSE in all cases. For the traffic dataset, the NRMSE incurred is lower when more models are output, and equal when the same number of model coefficients are output. This correlation between the number of models output and reduced NRMSE concurs with the results presented in Chapter 4.

Thus, in many cases a higher NRMSE is incurred when the storage used by the mutually reduced dataset is lower than the storage used by the independently reduced datasets. This result is consistent with the relationship between storage used and NRMSE incurred by  $k$ D-STR described in Chapter 4. However, in some cases the NRMSE incurred is equal or higher and the storage used by the reduced dataset is also higher. Specifically, this occurs for all features when  $\alpha \in \{0.1, 0.25\}$  and the traffic and rainfall datasets are mutually reduced, and the *temperature* feature when  $\alpha = 0.25$  and the traffic and air temperature datasets are mutually reduced, as well as when all three datasets are mutually reduced. In the latter two cases, the number of partitions in the mutually reduced dataset is higher than the total number of partitions in the independently reduced datasets, yet the NRMSE incurred for the *temperature* feature is higher.

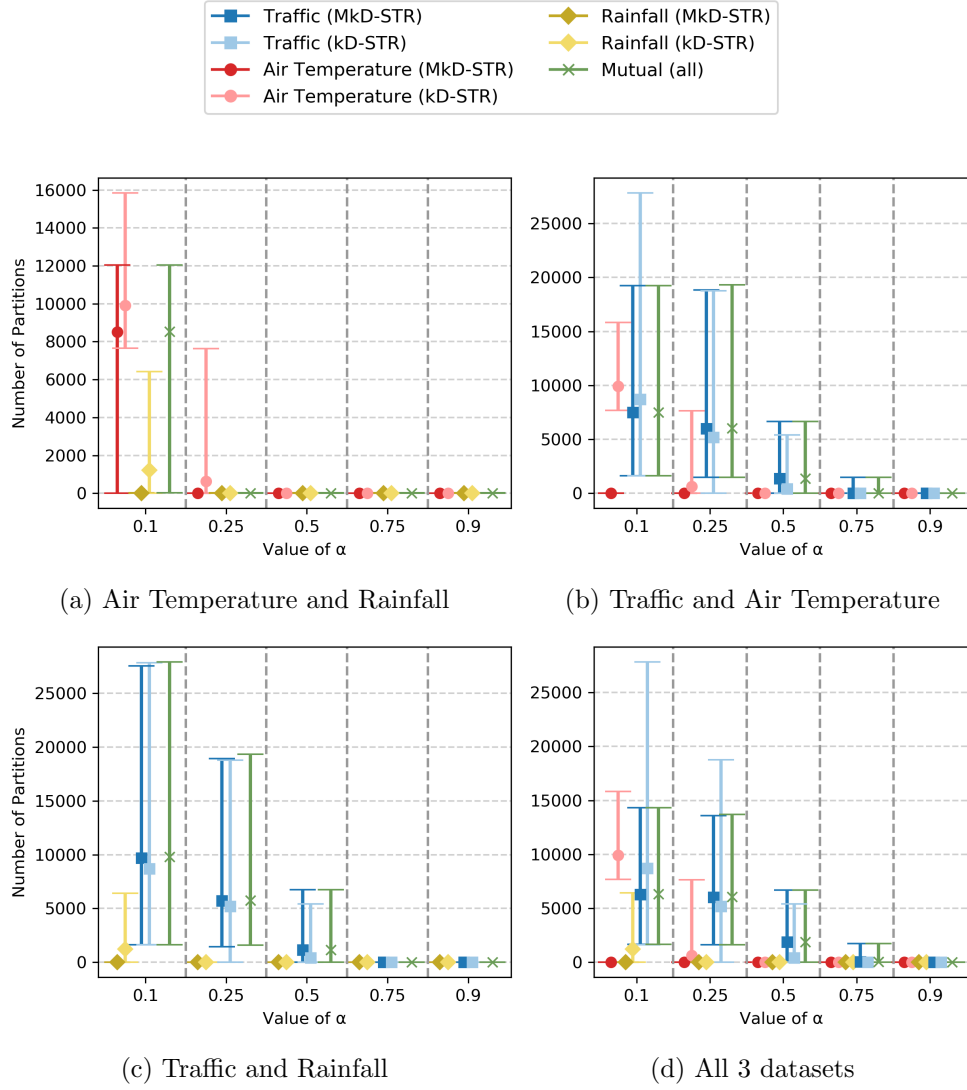


Figure 6.4: Number of models output by MkD-STR versus kD-STR. For each combination of datasets, the number of partitions (and models) output per dataset is denoted (*MkD-STR*), while the number of partitions (and models) output when each dataset is reduced independently is denoted (*kD-STR*). Total number of partitions output by MkD-STR is denoted *Mutual (all)*.

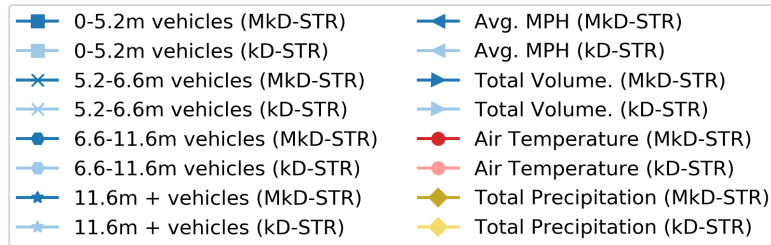
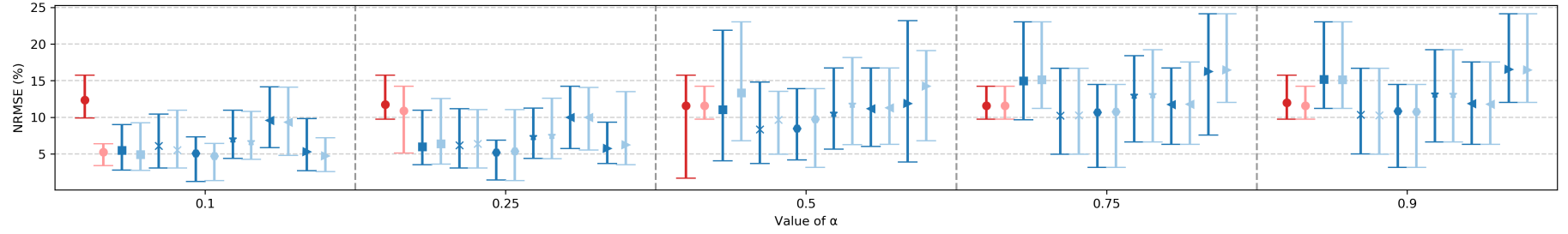
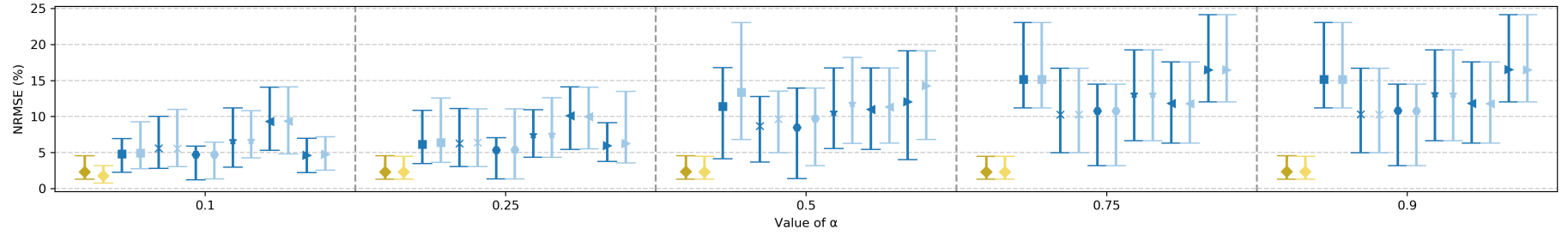


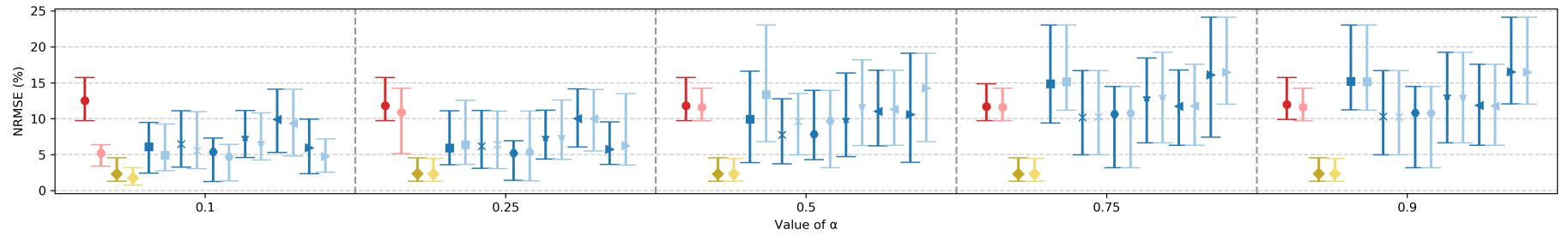
Figure 6.5: Legend used for Figure 6.6. Results are shown for each feature in the 3 datasets tested — results for the traffic dataset are shown in blue, results for the air temperature dataset are shown in red and results for the rainfall dataset are shown in yellow. Darker colours are used to denote results for MkD-STR and lighter colours are used to denote results for kD-STR.



(a) Traffic and Air Temperature



(b) Traffic and Rainfall



(c) All 3 datasets

Figure 6.6: NRMSE incurred by  $k$ D-STR versus  $Mk$ D-STR. The legend for this figure is shown in Figure 6.5.

### Effect on Running Time

In most cases, the time required to mutually reduce the datasets is higher than the time required to reduce any of the individual datasets. For example, when  $\alpha = 0.25$  and all three datasets are mutually reduced, the mean running time is 6,014 minutes. In contrast, the mean running times for individually reducing the traffic, air temperature and rainfall datasets are 4,066 minutes, 1,128 minutes and 699 minutes respectively. In all cases when  $\alpha > 0.1$ , the mean running time for mutually reducing the datasets is higher than the mean sum of reducing the datasets individually. However, in all cases when  $\alpha = 0.1$ , the mean running time for mutually reducing the datasets is lower than the mean sum of reducing the datasets individually. For example, the mean running time when  $\alpha = 0.1$  and all three datasets are mutually reduced is 3,208 minutes while the mean sum when the datasets are individually reduced is 10,726 minutes.

#### 6.1.4 Discussion

In this section, an adaptation of the  $k$ D-STR algorithm,  $Mk$ D-STR, was presented that overcomes the ambiguity of how to mutually reduce multiple spatio-temporal datasets. By mutually reducing two or more datasets, a set of common partitions is used to partition and model each of the datasets. This allows the direct comparison of partitions between the mutually reduced datasets, overcoming the many-to-many mapping between partitions when the datasets are reduced independently using  $k$ D-STR. The  $Mk$ D-STR algorithm uses the variability of the features in each dataset to inform the partitioning process, overcoming the incomparability of different features in the datasets.

In Section 6.1.3,  $Mk$ D-STR is shown to output fewer partitions than  $k$ D-STR when  $\alpha = 0.1$ . In this case, the increase in storage required to increase the number of partitions is not outweighed by the subsequent reduction in NRMSE, and so the number of partitions and models output is fewer than  $k$ D-STR. However, the results show that when  $\alpha \in \{0.25, 0.5\}$ ,  $Mk$ D-STR tends to output more partitions than  $k$ D-STR. In these cases, the increase in storage does not prevent more partitions being output. These results support Hypothesis H3.1. Similarly, Hypothesis H3.2 is also supported by these results. When the number of partitions output by the mutual reduction is lower than the sum of partitions in the independently reduced datasets, the NRMSE incurred increases. Similarly, when the number of partitions generated for a single dataset prior to the intersection step within the mutual reduction process is lower than the number in the individually reduced dataset, the NRMSE incurred is also lower. Conversely, when the number of partitions for a single dataset is higher, the NRMSE incurred is lower. Thus, both Hypotheses



H3.1 and H3.2 are supported by the results found in this section, and they demonstrate the utility of mutually reducing datasets while demonstrating the increased NRMSE that can occur when the partitioning of one dataset is dominated by another within the partitioning process.

## 6.2 $k$ D-STR: Distributed Spatio-Temporal Reduction

Increasing the number of datasets to be reduced, or increasing the quantity of data in a single dataset, has a significant impact on the running time of  $k$ D-STR. While  $k$ D-STR was shown to be effective at reducing spatio-temporal datasets in Chapter 4, it has an initial clustering complexity of  $\mathcal{O}(|D|^2)$  and, in the case of polynomial linear regression (PLR) modelling, a cost of  $\mathcal{O}((x + y^2|M|)|D|)$  per iteration. Furthermore, the initial clustering step of  $k$ D-STR requires  $\mathcal{O}(|D|^2)$  memory, and each iteration requires  $\mathcal{O}(|D| + y^2|M|)$  memory<sup>2</sup>. This complexity limits the number of instances that can be practically reduced using  $k$ D-STR, and this limit is quickly reached as the size of a dataset increases. To overcome this issue, the processing capacity of multiple computers can be used to reduce a dataset, thereby increasing the quantity of data that can be reduced.

Many frameworks for processing large datasets follow the MapReduce programming model which segments the data into portions for processing on independent compute nodes [39]. The process of segmenting the data and processing each segment on compute nodes is referred to as the *map* stage of MapReduce. After the map stage, the *reduce* stage processes the output of the map stage to generate a single result. For example, this result may be a single value or an updated dataset. In the context of  $k$ D-STR, multiple map and reduce stages can be combined to reduce a dataset. First, the clustering step of  $k$ D-STR can be performed using a distributed clustering algorithm, which segments and clusters the dataset in the feature space. Each compute node is allocated a segment of the feature space and is responsible for clustering the data within that segment. Second, the dataset is re-segmented in the spatio-temporal space and distributed again between the compute nodes. The compute nodes then iteratively reduce the data in rounds, utilising a designated *parent node* to control the reduction process.

However, adapting the  $k$ D-STR algorithm for the MapReduce paradigm presents an issue. After clustering the data, contiguous partitions of instances that belong to the same cluster must be found and these partitions may

---

<sup>2</sup>Here,  $|D|$  is the number of instances in a dataset  $D$ ,  $|M|$  is the number of models in the reduced dataset,  $x$  is the maximum number of instances that are adjacent to any single instance in  $D$ , and  $y$  is the maximum number of coefficients calculated for any model in  $M$ .

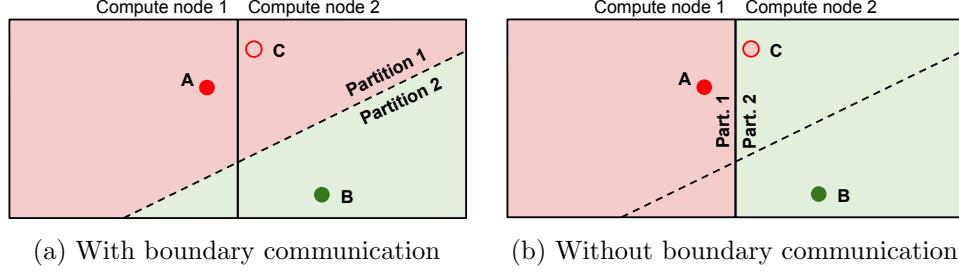


Figure 6.7: Effect of allowing partitions to span the boundary between compute nodes (with boundary communication) and not allowing partitions to span boundaries (without communication). Imputed instances may be less accurate when boundary communication is disabled.

span multiple compute nodes. While the process of discovering contiguous partitions can be distributed efficiently, the instances within a partition must be transferred to a single compute node to be modelled, incurring a significant communication overhead. A simple approach for solving this issue is to prevent partitions spanning multiple compute nodes, however this may increase the number of partitions created and increase the storage used by the reduced dataset. Furthermore, the imputation error of data at locations and times whose nearest instance belongs to a neighbouring compute node may increase. This problem is demonstrated in Figure 6.7. In both subfigures, instances A and B belong to neighbouring compute nodes, and the natural Voronoi boundary between them is shown as a dashed line. In subfigure (a) which allows partitions to span multiple compute nodes, the model of partition 1 is used to impute the feature values at location C. In subfigure (b) which does not allow partitions to span multiple compute nodes, the model of partition 2 is used to impute the feature values at location C. Thus, the value imputed in subfigure (b) may be less accurate than the value imputed in subfigure (a) as the nearest instance used to generate the partition’s model is further away. Therefore, an adaptation for the  $k$ D-STR algorithm is presented in this section that overcomes this issue, namely, Distributed  $k$ D-STR (D $k$ D-STR).

### 6.2.1 Adapting $k$ D-STR for Distributed Environments

The MapReduce computational model employs multiple computation nodes to execute an algorithm [39]. Each node applies the algorithm to a subset of the input data, and the results of the nodes are combined into a single output. A *parent node* controls the process and ensures that there is minimal overlap in the work carried out by each of the other compute nodes, referred to as *worker nodes*. The model consists of five steps:

1. **Ingest data:** Each instance in the input data is assigned a key  $k1$ , either at random or based on its index in the dataset. The parent node assigns

a  $k1$  key to each worker node, which read in the input instances labelled with their  $k1$  key. Note that many instances may be assigned the same  $k1$  key.

2. **Map:** Each worker node runs the user-defined  $Map()$  function on its data. This function re-labels each instance with a second key, referred to as  $k2$ . Again, many instances may be assigned the same  $k2$  key.
3. **Shuffle:** The worker nodes write the instances, labelled with their  $k2$  keys, back to temporary storage. The parent node then assigns a  $k2$  key value to each worker node, and the worker nodes read in the instances from temporary storage that are labelled with their  $k2$  key.
4. **Reduce:** Each computation node applies the  $Reduce()$  function to its portion of input data.
5. **Collect Results:** The parent node collects the  $Reduce()$  output from each worker node and computes a final output. This may be a single value, collection of values or updated values for each instance.

In the context of spatio-temporal data, the  $k1$  key for an instance may be an approximate indicator of its position in the input file. For example, if an input dataset  $D$  contains 1,000,000 instances, the  $k1$  key may be a label  $1, \dots, 100$ , thus each worker node would be assigned 10,000 instances after step 1. The subset of  $D$  belonging to a worker  $w$  is denoted  $D^w$ . The  $k2$  key of an instance may then be a function of its spatial location and time, thus  $Map()$  assigns all instances within the same spatial area and time period to a single computation node. Note that the step 1 is necessary as the distribution of locations and time periods in the data may not be known ahead of time, and so the parent node has to compute the spatio-temporal boundaries of each computation node between steps 1 and 2.

The  $kD$ -STR algorithm can be viewed as three separate processes. First, the input dataset is clustered in the feature space to create a hierarchical cluster tree. Second, homogeneous partitions of adjacent instances with respect to their cluster identifier are found in the spatio-temporal domain. Each of these partitions are then modelled using a predefined modelling technique, and a heuristic value for this reduction calculated over the set of partitions and models. Finally,  $kD$ -STR iteratively tests increasing the number of partitions in the reduced dataset and increasing the number of coefficients stored for each model.

In this section, an adaptation of these processes for distributed reduction is presented. Collectively, these give the Distributed  $kD$ -STR algorithm, referred to as  $DkD$ -STR. This algorithm first segments and distributes the

data in the feature space to compute a hierarchical clustering. Then, the algorithm redistributes the data between the worker nodes in the spatio-temporal domain for partitioning and reduction. Like  $k$ D-STR,  $Dk$ D-STR is an iterative algorithm that uses the parent node to control the reduction process.

### **Distributed Clustering**

To cluster a dataset, its instances are mapped to the worker nodes using their positions in the feature space. Then, the data is clustered in the feature space on each machine using a distributed hierarchical clustering algorithm. In literature, distributed algorithms have been presented for hierarchical clustering. Naive approaches use distributed data structures, such as the KD-tree as used by Woodley *et al.* [138], to approximate hierarchical clustering trees. Many alternative approaches focus on clustering with the single linkage function (minimum distance between any pair of instances in neighbouring clusters), finding the minimum spanning tree (MST) of the data which is equivalent to the single linkage hierarchical cluster tree [62]. Another alternative approach, the DHC algorithm, has been shown to enable hierarchical clustering in a distributed environment while permitting a range of linkage functions to be used, although the overhead of communication between compute nodes may limit the range of datasets that can be clustered [146]. A review of these techniques can be found in Appendix E.

Many distributed clustering methods use the MapReduce model to compute the cluster tree over the data, with the final tree stored on the parent node. In  $Dk$ D-STR, the data is then remapped to the worker nodes using the instances' locations in space and time, thereby redistributing the data using the spatio-temporal domain rather than the feature space. To do so, each worker node sends the list of unique sensors and timestamps in its subset of the dataset to the parent node. The parent node then assigns each worker node with a segment of the spatio-temporal space, and each instance is moved to the worker node responsible for its location in space and time. After the data is remapped according to the spatio-temporal domain, the parent node can broadcast the cluster boundaries for a given level of the cluster tree to each worker node, and the instances on each worker node can be labelled with their cluster identifier for that level of the cluster tree.

### **Distributed Partitioning**

After clustering the data in the feature space and redistributing the data between the worker nodes using the spatio-temporal domain, homogeneous partitions of instances belonging to the same cluster can be found. However,

partitioning the data using the  $k$ D-STR algorithm in a distributed manner introduces an inherent issue: if partitions are allowed to cross the boundaries between neighbouring worker nodes, significant communication overhead is incurred when moving the data onto a single node for modelling; yet, if partitions are prevented from crossing boundaries, the imputation of data from the reduced dataset may become less accurate. To overcome this issue,  $Dk$ D-STR employs a hybrid approach that allows partitions to cross worker node boundaries but not include instances on other nodes. That is, the spatial and temporal boundaries of the partitions are not clipped by the boundaries of the spatio-temporal domain applicable to each worker node, yet partitions may only include instances belonging to the same worker node.

To achieve this, each worker node requires knowledge of how the instances within its own subset of the dataset relate to the instances on its neighbour worker nodes. It must be able to create partitions whose bounds are equidistant between sensors (i.e. the Voronoi partitioning of space) in space and equidistant between timestamps in time, requiring significant communication between neighbouring worker nodes. However, since the parent node is aware of all sensors and timestamps in the data, it may provide this dissection of space and time to each of the worker nodes. That is, the parent node can compute the Voronoi partitioning of the sensors in the spatial domain and provide the mid-point between timestamps in the temporal domain. Each worker can then partition the subset of instances within its division of the spatio-temporal domain using the same process described in Section 4.2.1 along with the Voronoi and timestamp partitioning provided by the parent node. This results in partitions that span the boundaries between worker nodes as shown in Figure 6.7a without requiring instances be communicated between worker nodes for modelling.

### Iterative Reduction Steps

To perform the iterative steps of  $k$ D-STR, the  $Dk$ D-STR algorithm must calculate the heuristic value in a distributed manner. The storage and error metrics used in the heuristics (i.e. 4.3, Equations 4.7 and 5.6) require knowledge of the storage required to store the set of partitions and models, as well as the NRMSE incurred when reconstructing the original instances or imputed features. Since these metrics can be decomposed into calculations over each partition and model in the reduced dataset, they can be partially calculated on each worker node and these partial results combined by the parent node. Thus, each worker node calculates the storage used and NRMSE incurred when the number of partitions is increased and the complexity of a model is increased. The parent node then calculates the heuristic value for each of

these possibilities and instructs the appropriate worker node to implement the change that minimised the heuristic value. When the heuristic cannot be minimised further, each worker node outputs its partitions and models, and the combination of these outputs yields the reduced dataset.

Specifically, for each possible increase in partitions and model complexities, the storage values sent from each worker node to the parent node are:

$$\text{storage}_w(D^w) = |D^w| \cdot (k + |F|) \quad (6.2)$$

$$\text{storage}_w(\langle P^w, M^w \rangle) = \sum_{i=1}^{|P^w|} (|b_i| \cdot (k - 1) + 2) + \sum_{j=1}^{|M^w|} |m_j| \quad (6.3)$$

where  $D^w$  is the subset of dataset  $D$  processed on worker node  $w$ , and  $P^w$  and  $M^w$  are the partitions and models output by  $w$  respectively. The storage ratio (i.e. Equation 4.7) calculated on the parent node is then:

$$q(D, \langle P, M \rangle) = \frac{\sum_{w \in W} \text{storage}_w(D^w)}{\sum_{w \in W} \text{storage}_w(\langle P^w, M^w \rangle)} \quad (6.4)$$

where  $W$  is the set of worker nodes, and  $P$  and  $M$  are the sets of partitions and models respectively collected from the set of worker nodes. The NRMSE calculation performed on each worker node for a given feature  $f$  is:

$$\psi_w(D^w, \hat{D}^w, f) = \sum_{d_{s,t} \in D^w} (d_{s,t}^f - \hat{d}_{s,t}^f)^2 \quad (6.5)$$

and the NRMSE (i.e. Equation 4.3) calculated on the parent node is:

$$e_{\text{NRMSE}}(D, \hat{D}, f) = \frac{\psi(D, \hat{D}, f)}{\text{range}(f)} \quad (6.6)$$

where,

$$\psi(D, \hat{D}, f) = \sqrt{\frac{\sum_{w \in W} \psi_w(D^w, \hat{D}^w, f)}{|D|}}$$

Using these metrics, the parent node can calculate the heuristic function that controls the reduction process. On each iteration of DkD-STR, each worker node tests the change in storage and NRMSE incurred when the number of partitions is increased (using the cluster definitions provided by the parent node), and the change when the complexity of a model changes. Conceptually, this results in each worker sending  $1 + |M^w|$  storage and NRMSE values to the parent node on each iteration. However, when the change that minimises the heuristic value most is a model complexity increase on one worker node, the values reported by the other worker nodes will not change. Therefore, in

practice fewer results will be sent from the worker nodes to the parent node for some iterations of the reduction process. Note that when the number of partitions is increased, the results for every worker node become invalidated and so  $1 + |M^w|$  storage and NRMSE values have to be sent by each worker node on the next iteration. Furthermore, if the complexity of a model on a worker node is increased on one iteration, increasing the same model's complexity again may be the best move on the next iteration. Thus, it would be improper to allow more than one worker node to increase the complexity of a model on each iteration as doing so may unnecessarily increase the complexity of models on some worker nodes.

### Summary of Distributed Data Reduction

The  $DkD$ -STR algorithm uses the MapReduce paradigm for distributed computation. One or more worker nodes are used to reduce segments of the input dataset, and are controlled by a parent node. The steps of the  $DkD$ -STR algorithm are listed below, and these correspond to the steps shown in Figure 6.8. In this overview, the reduction of a single dataset is presented, although the alternative heuristic and partitioning schemes presented in this chapter and Chapter 5 may be used in other scenarios.

Like  $kD$ -STR,  $DkD$ -STR requires an input dataset, value for the parameter  $\alpha$ , and a chosen modelling technique. The output of  $DkD$ -STR is a set of partitions and models,  $\langle P, M \rangle$ . The steps of  $DkD$ -STR for a single dataset  $D$  are:

1. **Data clustered in feature space:** The dataset  $D$  is distributed between the worker nodes using the feature values as key  $k$ . Then, the data is clustered using a distributed hierarchical clustering technique. The resulting tree of cluster definitions is stored on the parent node.
2. **Spatial and temporal domains segmented:** Each worker node sends its list of unique sensor locations and instance timestamps to the parent node. The parent node then segments the spatio-temporal domain according to the number of compute nodes, and assigns each worker node a segment. Using the lists of sensor locations and timestamps, the parent node computes a Voronoi partitioning (equidistant) partitioning of space and time. This partitioning is sent to the worker nodes.
3. **Initial reduction step:** The dataset is redistributed between the worker nodes using the segmentation computed by the parent process. The first reduction step is performed: each worker node forms one partition of all of its instances using the distributed partitioning method presented in this chapter; then, each worker models the data in the partition using

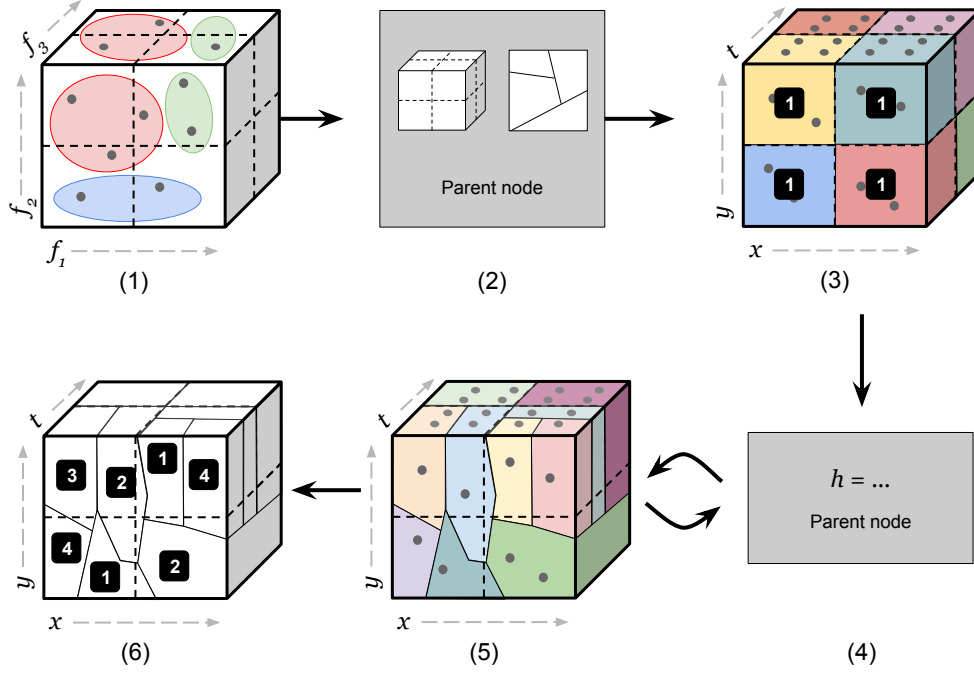


Figure 6.8: Overview of DkD-STR with 8 worker nodes: (1) data is clustered in the feature space, (2) spatial and temporal domains are segmented by parent node, (3) initial reduction step yields 1 partition and model coefficient per worker node, (4) parent node calculates the heuristic value of step 3, (5) partitions on each node are iteratively decomposed and modelled, (6) final set of partitions and models is output.

the predefined modelling technique, and calculates the storage used and error incurred. These values are then sent to the parent node.

4. **Heuristic calculation:** The parent node collects the storage and error values sent by each worker node and combines them (Equations 6.4 and 6.6) to calculate the heuristic value.
5. **Iterative reduction steps:** The following steps are repeated until the heuristic cannot be minimised further.
  - (a) **Increase the complexity of a model:** Each worker node increases the complexity of each of its models in turn, and calculates its error and storage values after each increase. This yields a  $|M^w|$ -length vector of storage values and  $|M^w|$ -length vector of error values for each worker mode.
  - (b) **Increase number of partitions:** Each worker node increases the number of partitions by increasing the number of clusters by 1. Each of the new partitions is modelled, yielding a single storage value and single error value for the worker node.



- (c) **Parent node decides:** Each worker node sends the results of increasing the complexity of each model and increasing the number of partitions to the parent node. This results in  $|M| + 1$  possible changes, and the parent node calculates the heuristic value for each. The change that minimises the heuristic is implemented on its worker node.

6. **Reduced dataset output:** When the heuristic cannot be minimised further, the reduction is complete. Each worker node outputs its set of partitions and models, and the union of these sets is the reduced dataset  $\langle P, M \rangle$ .

### 6.2.2 Experimental Evaluation Methodology

To evaluate the impact of reducing a dataset in a distributed setting, the datasets presented in Chapter 3 are reduced using DkD-STR. The distributed environment was simulated using separate threads on a multicore processor. During each iteration, the maximum time used by any single thread was used as the time taken for that iteration. In these results, the impact of allowing partitions to cross worker node boundaries is tested, as well as the speedup achieved by DkD-STR over  $k$ D-STR. While these results do not demonstrate the full speedup possible when reducing datasets in a distributed setting, they are indicative of the speedup achievable in a naive implementation. Specifically, 2 hypotheses are tested:

- H4.1 *Increasing the number of worker nodes used yields an approximately superlinear reduction in the time taken to reduce the data.*

Since  $k$ D-STR is a  $\mathcal{O}(|D|^2)$  algorithm, distributing the algorithm should give a greater than linear speedup. For example, using 2 processors instead of 1 should reduce the processing time by more than 50%. However, this assumes the cost of communication is negligible and that the distribution of instances between worker nodes is uniform. Since this is not guaranteed, and the implementation of DkD-STR is not optimised, the speedup achieved may be significant as the number of worker nodes increases from 1, but become negligible as the number of worker nodes increases further.

- H4.2 *The partitioning technique used by DkD-STR decreases the error incurred when imputing instances compared to independently partitioning the data on each worker node.*

To prevent a significant communication overhead when partitioning and modelling the data, a naive approach would prevent the partitions on each worker node from crossing the boundary between neighbouring nodes. However, as shown in Figure 6.7, this may lead to increased error

when imputing data from the reduced dataset. Thus, the partitioning technique used by D $k$ D-STR, which overcomes this issue, incurs a lower imputation NRMSE compared to the naive approach which does not permit partitions to cross the boundaries between neighbouring nodes.

To test the speedup achieved by increasing the number of worker nodes, 5 sets of worker nodes are simulated and tested. For each of the spatial and temporal dimensions (i.e. latitude, longitude and time), the number of worker nodes is increased from 1 to 5 yielding 5 pool sizes of worker nodes: (i)  $1 \times 1 \times 1 = 1$  node, (ii)  $2 \times 2 \times 2 = 8$  nodes, (iii)  $3 \times 3 \times 3 = 27$  nodes, (iv)  $4 \times 4 \times 4 = 64$  nodes, (v)  $5 \times 5 \times 5 = 125$  nodes. As these experiments focus on the rate of speedup achieved by increasing the number of worker nodes to reduce a dataset, PLR-P modelling is again used as a consistent modelling technique to demonstrate the rate of speedup achieved by increasing the number of worker nodes used. The PLR-P technique is used as it provides an average case reduction compared to DCT and DTR modelling (as shown in Figure 4.7), and modelling on partitions is shown to yield a lower NRMSE in many cases in Section 4.4. As shown in Chapter 4, the NRMSE incurred and storage reduction achieved are directly correlated with the number of partitions in the reduced dataset. Thus, results for the other modelling techniques presented in Chapter 4 can be inferred by considering the speedup achieved for PLR-P modelling in this section with the results presented in Section 4.4.

### 6.2.3 Results

In this subsection, the results of the experiments described in Section 6.2.2 are presented.

#### Effect on Processing Time

For all three datasets, increasing the number of worker nodes decreases the processing time required to reduce the dataset, as shown in Figure 6.9a. For example, when  $\alpha = 0.1$ , the mean time taken to reduce the traffic dataset falls from 234,021 seconds to 433 seconds. However, as the number of nodes increases, the rate of decrease in processing time also decreases. This is correlated with the maximum number of instances on a single node, with the processor reducing the most instances taking the longest to process each iteration. As shown in Table 6.2, increasing the number of processors from 1 to 125 for the traffic dataset decreases the maximum number of instances on a partition from 149,495 to 5,818. In contrast, a balanced distribution would place 1,196 instances on each node. This sublinear decrease in the maximum number of instances on any node is correlated with the a sublinear decrease in the time taken as the number of worker nodes increases.

Table 6.2: Maximum number of instances on any worker node, averaged over all data samples for each dataset.

	1x1x1	2x2x2	3x3x3	4x4x4	5x5x5
Traffic	149,495	32,757	15,957	8,755	5,818
Air Temperature	232,043	39,230	15,131	6,847	4,020
Rainfall	187,508	31,511	12,073	5,511	3,230

As shown in Figure 6.9b, the number of partitions stored for a dataset is approximately equal to the number of worker nodes used to reduce the dataset when 1 cluster is used to partition the dataset<sup>3</sup>. That is, since partitions do not include instances from multiple worker nodes, the number of partitions increases as the number of worker nodes increases. In some cases when the number of clusters used is greater than 1, the number of partitions output is affected more by the variability of the data than the number of worker nodes used. In Figure 6.9b, this is particularly noticeable for the traffic and air temperature datasets when  $\alpha = 0.1$ ; here, the number of partitions does not increase noticeably as the number of worker nodes increases. In other cases, however, increasing the number of worker nodes increases the number of partitions enough that the required storage is too great and only 1 cluster is used to partition the data. This is particularly noticeable for the rainfall dataset when  $\alpha = 0.1$ , and the air temperature dataset when  $\alpha = 0.25$ . In both cases, increasing the number of worker nodes results in a drop in the number of partitions output, as the number of clusters used in the partitioning process decreases from 2 to 1. Thus, for datasets that are highly varied in space and time, increasing the number of worker nodes does not affect the number of partitions output. However, for datasets with a low variability in space and time, increasing the number of worker nodes may reduce the number of partitions output. As discovered in Chapter 4, the storage used and reconstruction error incurred are correlated with the number of partitions in the reduced dataset (as shown in Appendix F).

### Comparison of D $k$ D-STR Partitioning versus Naive Partitioning

To test Hypothesis H4.2, each of the datasets are reduced using  $2 \times 2 \times 2$  processors with the ability of partitions to cross worker node boundaries enabled and disabled. Prior to reduction, instances that are close to the worker node boundaries are removed and the datasets reduced without these instances. For the datasets presented in Chapter 3, larger pools of processors (i.e.  $3 \times 3 \times 3$

<sup>3</sup>In some cases, missing instances cause a separation in the spatial or temporal domain that requires 2 or more partitions to capture. Thus, the number of partitions stored is sometimes marginally higher than the number of worker nodes.

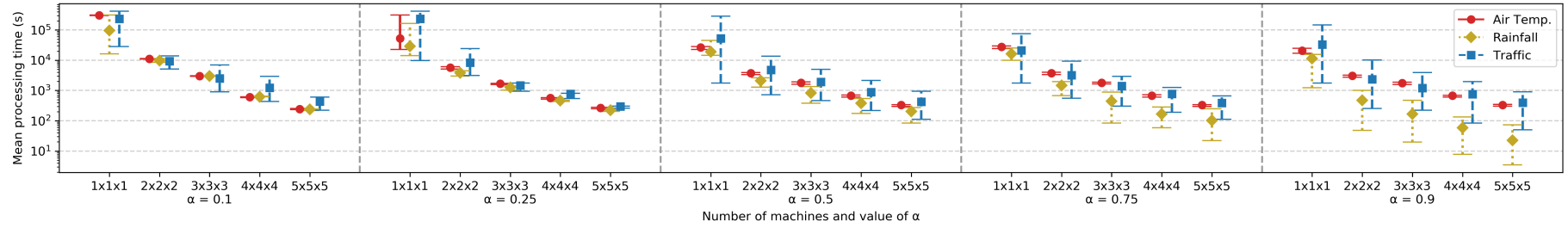
and larger) result in a majority of instances being close to node boundaries and therefore removed, thus only the pool size of  $2 \times 2 \times 2$  processors is used. The error incurred when imputing the withheld instances from the reduced datasets is then measured, and these results can be seen in Figure 6.10.

For the rainfall and traffic datasets, preventing partitions from crossing node boundaries increases the NRMSE incurred when imputing the withheld instances. In some cases, notably the rainfall dataset when  $\alpha \geq 0.1$ , preventing partitions crossing boundaries does not change the maximum NRMSE incurred, but does increase the mean error. For the air temperature features, the mean error decreases when partitions do not span boundaries and  $\alpha \in \{0.01, 0.25\}$ . Furthermore, the maximum error decreases when  $\alpha \geq 0.1$ .

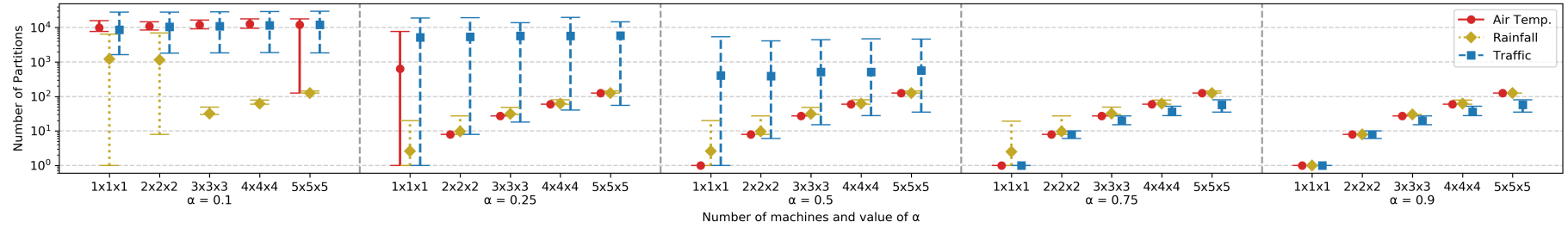
#### 6.2.4 Discussion

In this section, the  $DkD$ -STR adaptation of the  $kD$ -STR algorithm has been demonstrated to speed up the reduction process. In the example of reducing the traffic dataset when  $\alpha = 0.1$ , increasing the number of worker nodes from 1 to 4 reduces the average time taken from 65 hours to 2.5 hours. However, as the number of nodes increases further the rate of decrease in processing time diminishes as the maximum number of instances on a single node decreases sublinearly. These results show a significant reduction in reduction time can be achieved using  $DkD$ -STR, and we may infer this increases the maximum dataset size that can be reduced. Furthermore, these results support Hypothesis H4.1, although they also show the rate of speedup achieved is correlated with the maximum number of instances on a worker node rather than the number of worker nodes used overall.

Hypothesis H4.2 is only partially supported by these results. In some cases, the imputation error is reduced by allowing partitions to span the boundaries between worker nodes, yet in others this is not so. This mix of results may be explained by a second source of error in the reduced dataset. Since the partitioning process is stochastic in nature, the set of instances belonging to each partition may change between reductions. Therefore, each partition may have a different model when reducing the same dataset multiple times. This may lead to more accurate imputations for the same withheld instances in some cases, and less accurate in others. The inconsistency between results observed for each dataset may therefore be a result of the stochastic nature of the partitioning and modelling process.



(a) Time taken to reduce dataset



(b) Number of partitions in reduced dataset

Figure 6.9: Time taken and number of partitions output when reducing the datasets presented in Chapter 3 using DkD-STR

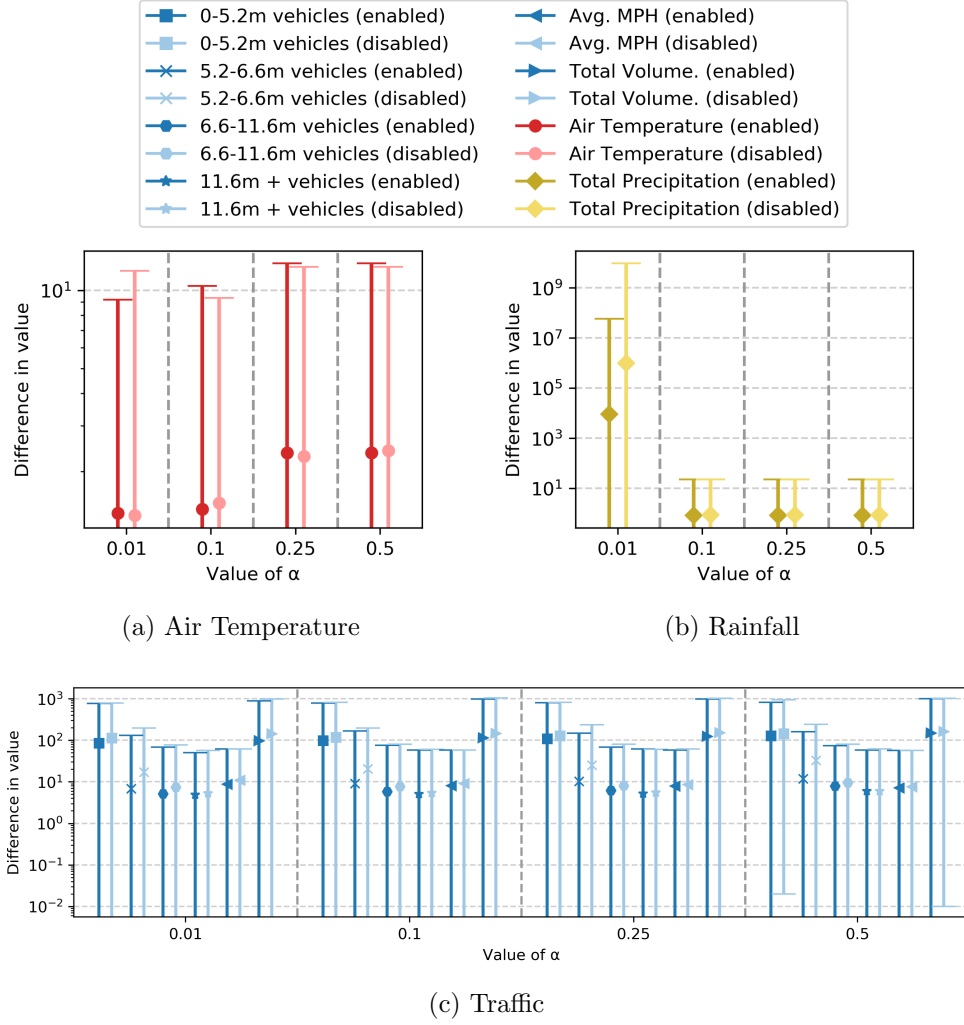


Figure 6.10: Imputation error incurred DkD-STR with the ability for partitions to span node boundaries enabled and disabled. When  $\alpha > 0.5$ , only 1 partition is stored per worker node, thus the error incurred is unaffected by the ability of partitions to span boundaries.

### 6.3 Impact of Errors and Missing Instances in Input Datasets

When processing and analysing real-world datasets, issues can be encountered such as missing instances and erroneous data. For example, systematic errors, noise and missing data can be problematic when calculating statistical measures and clustering the data. However, by understanding the robustness of these methods to such issues, the effects of these issues can be managed. In literature, empirical analyses of algorithms are used to evaluate the robustness of methods, as discussed in Section 2.5. Therefore, this section presents an empirical analysis of the effects of erroneous data on  $k$ D-STR. The effect of commonly occurring

errors are studied for road traffic, air temperature and rainfall data, as identified by analysts who use these datasets within the transportation domain. Namely, the impact of errors in location and time, noisy feature values and missing data on the reduced output of  $k$ D-STR are measured. The exact impact of these issues may differ for other datasets, however the conclusions of this analysis can be used to inform the use of  $k$ D-STR with other data. More specifically, as the amount of error or missing data increases, this analysis explores:

1. How the number of partitions and models changes, as well as the number of model coefficients.
2. How the error incurred when reconstructing the input (erroneous) dataset from the reduced dataset changes, how the error between the clean dataset and the reconstructed dataset changes, and how the distribution of these errors compares against the distribution of error added in the input (erroneous) dataset.
3. How the storage used by the reduced dataset changes.
4. How imputed instances change (that is, instances that were not present in the erroneous input dataset).

Four types of error are studied in this section as described in Section 6.3.1. In Section 6.3.2, the experimental setup and datasets used are discussed. Finally, the results of this study are presented in Section 6.3.3.

### 6.3.1 Sources of Error and Missing Data

Typically, issues encountered with sensor data are incurred during sensing (as discussed in Chapter 2.5). Unlike communications and storage infrastructure, which often use hardware and software checks to validate that data has not changed, it can be difficult to validate that sensor values are accurate and not erroneous. It is therefore useful to assume that sensor data may be erroneous and to study the impact of such errors on the algorithms used to process and analyse the data. In many studies found in literature on the impact of erroneous spatio-temporal data, three issues are repeatedly identified as commonly occurring. These are error in space and time, noise in the feature values, and missing data. Each of these are discussed further in this section.

#### Errors in Location of Sensors

In literature, the sensitivity of spatio-temporal methods to error in location is often tested through spatial perturbations of the data. As described in Section 2.5.1, many studies analyse the increase in error incurred by spatio-temporal

methods when the location of sensors are moved. To achieve this, studies perturb the data using distributions that are tailored to the dataset being used, reflecting the common sources of location error for each dataset. For example, some studies move the location of a sensor in the data by a distance chosen from an empirical distribution in a heading chosen from a uniform distribution [40, 87].

However, the locations of sensors used to collect spatio-temporal data are often recorded using GPS devices. In literature, the spatial error incurred by GPS devices is shown to exhibit bivariate normal distributions that are independent in the  $x$  and  $y$  directions, with a mean value of 0 [63, 102]. As described in Section 2.5.1, 95% of GPS locations recorded are accurate to within approximately 3 m of the true location [63, 102]. In transport related datasets, the locations of road-side sensors are often recorded as the unique identifier of the nearest road segment or *link* to the sensor. Since the maximum length of any section in the traffic datasets is 3086 m, it can reasonably assumed that the maximum location error of any roadside sensor may be 3086 m. Thus, the locations of instances in the traffic, air temperature and rainfall datasets may be assumed to be incorrect by up to 2.5 m if GPS devices are used to record the locations of sensors, and up to 3086 m if the sensor locations are referenced by their road section location instead.

Therefore, an analysis of the effects of spatial location error was performed on the  $k$ D-STR algorithm. Since the sensors used to record the datasets presented in Chapter 6 are stationary and do not move, we can reasonably assume that the location of each sensor is established when the sensor is installed, and does not change thereafter. Each sensor is perturbed in space by an amount drawn from a normal distribution in each spatial dimension, using a similar approach to those presented in literature and reflecting the known distributions of GPS errors [63, 102]. That is, a normal distribution is used for each dimension with a mean value of 0 and a range of standard deviation values, namely 1.5 m (corresponding to the 3 m 95% percentile of GPS inaccuracy found in literature), 350 m, 700 m, 1029 m (corresponding to a 95% percentile of 3086 m), and 2000 m.

### **Errors in Timestamps of Instances**

Similar analyses to those discussed for the spatial domain have been found in literature for the temporal domain, as discussed in Section 2.5.2. In some cases, empirical distributions are used to estimate the distribution of error between the recorded time for each instance and the true times the instances were recorded at. In others, domain knowledge is used to estimate the distribution of temporal error. For example, in their analysis of Dengue Fever, Delmelle



*et al.* estimated the maximum delay between a person first experiencing symptoms and visiting their doctor is 5 days [40]. Therefore, to test the effect of temporal error, a temporal offset was applied to each instance, chosen from a Gaussian distribution with mean 0 and variance of 1.68 days (corresponding to a maximum value of 5 days using the *empirical rule* of 3 standard deviations).

In the datasets presented in Chapter 3, the recorded timestamp for each instance is believed to be the correct time the instance was recorded. Therefore, no empirical distributions of error are known from which temporal perturbations can be drawn. However, a reasonable assumption can be made about the maximum temporal error in the dataset in a similar approach to that used by Malizia and Delmelle *et al.*. In many systems, time synchronisation is performed using protocols such as the Network Time Protocol (NTP), which limits the maximum time between resynchronisations as 1024 seconds (s) and the drift to less than 100 milliseconds (ms) [81]. Since the minimum time between instances in the datasets presented in Chapter 3 is 15 minutes, it can be assumed that the synchronisation error is negligible and instead the clock at each sensor may be incorrect by a fixed offset of up to 15 minutes.

To test the effect of temporal offsets on *kD-STR*, a similar analysis to that performed for the spatial domain is performed for the temporal domain. It may be assumed that, in a worst-case but reasonable scenario, each sensor is impacted by a source of temporal error that affects every instance recorded at that sensor identically. Since the air temperature and rainfall datasets contain instances recorded at 1-hour intervals, and the traffic datasets contain instances recorded at 15-minute intervals, the maximum delay can be assumed to be 1 hour and 15 minutes respectively, otherwise the server or network would detect the reporting error. Using the assumption that temporal error in sensor networks exhibits a Gaussian distribution with mean value of 0 [65], and the empirical rule also used by Delmelle *et al.* [40], the maximum errors of 1 hour and 15 minutes yield a maximum standard deviation of 20 minutes for the air temperature and rainfall datasets, and 5 minutes for the traffic dataset. Thus, a temporal offset chosen from a Gaussian distribution is applied to each sensor and all instances recorded by that sensor offset by the same length of time. The distributions used for the air temperature and rainfall datasets have a mean value of 0 and standard deviations of 5 minutes (mins), 10 mins, 15 mins, 20 mins and 40 mins. The distributions used for the traffic dataset has a mean value of 0 and standard deviations of 1.25 mins, 2.5 mins, 3.75 mins, 5 mins and 10 mins.

## Noise in Feature Values

To study the impact of measurement error on a spatio-temporal method, noise is added to the input dataset and the difference in the method's output is measured. In many cases seen in literature (see Section 2.5.3), noise chosen from a normal distribution is added to the feature values of each instance. For example, in the hydrological domain, Sivakumar *et al.* added normally distributed noise to a rainfall dataset corresponding to 0.04, 0.08 and 0.16 standard deviations of the original features to test noise removal techniques [117]. In this case, the parameters chosen for the noise distribution were informed by the range of feature values that are likely to be observed in the data. In other studies, analysts deliberately use increasing amounts of noise that go beyond the feature values that are likely to be seen in a specific domain. For example, Quirós *et al.* injected Gaussian noise to fMRI data with three variance values to test the robustness and accuracy of modelling techniques in the presence of noise [101]. Three standard deviations were tested with the maximum standard deviation tested equal to the maximum value possible in the feature. In their analysis of the k-Nearest Neighbours (kNN) imputation technique, Cheng *et al.* added up to 2 standard deviations of the features' distribution [30]. This approach allowed them to analyse the impact of highly erroneous feature values on the kNN imputation technique for a range of values of  $k$ .

The aim of this study is to analyse the impact of feature errors on the reduction of datasets by  $k$ D-STR. To do so, the same approach taken by the methods found in literature was used, and noise chosen from a normal distribution was added to each instance. The noise incident on networks of sensors, such as air temperature and rainfall sensors, has been shown to have a Gaussian distribution with a mean value of 0 (according to central limit theorem) [65, 83, 147]. The worst case found in literature (used by Cheng *et al.*) is used as the upper bound on the distributions of noise, giving a maximum standard deviation of the error distribution equal to 2 standard deviations of the clean features. Thus, for each feature in the datasets, noise is added to the feature values chosen from a Gaussian distribution with mean value of 0 and standard deviations equal to 0.1, 0.5, 1.0, 1.5 and 2.0 standard deviations of the original features.

## Sources of Missing Instances

As described in Section 2.5.4, multiple studies in literature analyse the impact of missing data on imputation and modelling methods. For example, Chen *et al.* studied the effect on imputation methods of removing increased numbers of instances from a road traffic dataset [29]. They tested both missing at

random (MAR) by removing between 10% and 50% of instances, and missing not at random (MNAR), which reflected a number of sensors going ‘offline’ for amounts of time drawn from a distribution. In both cases they reported the MAPE and RMSE imputation error metrics for a set of instances that were withheld for ground truth. A similar study in the transport domain was performed by Rodrigues *et al.* who removed 10%, 25%, 50% and 75% of the data at random, and Chuan *et al.* in the hydrology domain, who removed 5%, 10%, 15%, 20%, 25% and 30% of the instances in a rainfall dataset to test to effect on imputation algorithms [34, 105]. In their analysis of the kNN technique in hydrology (where  $k = 10$ ), Lebecherel *et al.* tested the impact of removing all sensors within 10 km to 200 km of a withheld sensor being imputed, as well as removing between 10% and 90% of the sensors at random [77]. The distances chosen were selected by considering the mean distance between any sensor and its 10 nearest neighbours in the dataset.

The aim of this study is to test the effect of missing data rates on the reduced datasets output by  $k$ D-STR. While all of the analyses found in literature follow a similar approach, the approach taken by Chen *et al.* specifically tests both MAR and MNAR scenarios. Since similar issues can arise in transport network, air temperature and rainfall datasets, this procedure is followed in this study. Increasing quantities of data, between 10% and 50% of instances, are removed at random and the effect on  $k$ D-STR output measured. Then, increasing quantities of data are removed again from the clean dataset by simulating ‘periods of silence’ for each sensor. To do so, a sensor and day are chosen at random and all instances recorded on that day by that sensor are removed. This process is repeated until the desired number of instances are removed — again, the MNAR rate ranges between 10% and 50%.

### 6.3.2 Experimental Evaluation Methodology

The sources of error and missing data described in Section 6.3.1 can occur in real-world data, and so it is necessary to understand the effect they have on reduced datasets. To do so, the datasets presented in Chapter 3 are perturbed by moving the locations of instances in space and time, adding noise and removing instances. The *unclean* (perturbed) datasets are then reduced using  $k$ D-STR and the reduced forms compared with the reduction of the clean datasets. Furthermore, by withholding portions of the data in a 10-fold cross-validation manner, the imputation error of both the clean and unclean reduced datasets can be compared.

As the amount of perturbation and missing data increases, the metrics described at the beginning of this section are evaluated. Here, the term perturbation is used to refer to the shifting of instances in time and space, and

the addition of noise to each instance’s feature values. Furthermore, missing data is used to refer to the removal of instances, both at random and not at random. For each of the perturbations, the following hypotheses are tested:

H5.1 *Increasing the amount of perturbation increases the variance of the data, thereby increasing the number of partitions in the reduced datasets.*

As the amount of error in the data increases, similar instances may move further apart, and instances that are close in space and time will become more dissimilar. As a result, the variance of the data will increase over space and time. Overall, the number of partitions may also increase as near instances are grouped into different clusters in the feature space.

H5.2 *The modelling process reduces the impact of the perturbations.*

As the amount of error increases, the variance within each partition also increases. As a result, the modelling process is unable to capture the variance of the unclean data as accurately as the clean data. Thus, the modelling process *smoothens* out the perturbations in the data, and the reconstructed data has a lower variance than the input unclean data. Furthermore, the reconstruction error of the unclean data increases.

H5.3 *Increasing the amount of perturbation increases the imputation error of withheld data.*

As a result of H5.2, the models stored diverge from the models stored in the reduced clean dataset. Thus, the imputation error of withheld clean instances increases.

For the two missing data experiments, the following hypotheses are tested:

H5.4 *Increasing the amount of missing data decreases the number of partitions output.*

Though removing instances may not change the variation of the data over space and time, some non-adjacent instances that belong to the same cluster in the clean dataset may become adjacent in the unclean dataset. As a result, the number of partitions decreases as the number of missing instances increases.

H5.5 *Increasing the amount of missing data increases the imputation error of withheld data.*

As more instances are removed from the unclean input dataset, the distance to the nearest instances for a withheld validation increases. As a result, the imputation error of the withheld validation instance increases.

Using the modelling techniques and values for  $\alpha$  used in Chapters 4—6.2, the parameter space to be tested for each of the four sources of error includes

5 standard deviation values, 10-fold cross validation, 3 modelling techniques (PLR, DCT and DTR), modelling on partitions and clusters, 5 values for the parameter  $\alpha$  (0.1, 0.25, 0.5, 0.75, 0.9), and 84 samples of data (12 1-month samples of air temperature and rainfall, and 12 samples of traffic data for 7 roads). Testing the entirety of this parameter space is impractical, and so a few subsets of the parameter space are tested instead. These results are presented and described in Section 6.3.3, and the results for untested areas of the parameter space can be inferred using both this analysis and the results presented in Section 4.4. Furthermore, four samples of data from the datasets presented in Chapter 3 are used: (i) A30 sample from August 2017, (ii) M1 sample from October 2017, (iii) air temperature from October 2017, (iv) rainfall from September 2017. These samples are chosen as they are the most stationary samples for the datasets in space and time, thus the imputation error incurred is most likely to be caused by the perturbations. Furthermore, these samples are generally representative of the distributions of all of the samples for the traffic, air temperature and rainfall datasets. Further discussion on the selection of these datasets can be found in Appendix G.

### 6.3.3 Results

Though the full parameter space of perturbations is too large to test, an initial analysis is used for  $\alpha = 0.1$  and all three modelling techniques, both modelling on partitions and clusters. From these results, the difference between modelling on partitions and clusters is shown to be the same as the results presented in Chapter 4, and so only modelling on partitions is tested to reduce the testing parameter space. The results of these perturbations are presented in this section. Note that when discussing the effects of noise, the standard deviations of the noise distribution are calculated as multiples of the standard deviation of the original features. To avoid ambiguity, the term *standard deviation* is used to refer to the standard deviation of the noise distribution, and the unit *SD* is used to refer to 1 standard deviation of the original features.

#### Effect on Variance

For each dataset, the variance of the perturbed features increases as the standard deviation of the distribution of noise increases. For example, the variance of the *Total Volume* feature of the A30 dataset increases from 21,130 to 21,336 when a standard deviation of 0.1 SD is used, and to 105,365 when 2.0 SD is used. These values can be seen in Appendix H.1.

In all cases, reducing the noisy data using *kD-STR* reduces the variance in the reconstructed instances. Thus, reducing the data reduces the impact of noise in the data as not all of the variance is captured by the partition models.

This is true for all modelling techniques and both modelling on clusters and partitions. When  $\alpha = 0.1$ , the variance in the reconstructed traffic and air temperature datasets increases as the standard deviation of noise increases. Again, the variance is lower than the input noisy data in all cases. In some cases for the rainfall dataset when  $\alpha = 0.1$ , increasing the standard deviation of noise increases the variance in the reconstructed dataset. However, in others cases, the variance decreases. For example, when DTR-P modelling is used, the variance in the reconstructed rainfall dataset increases from 0.081 to 0.089 as the standard deviation increases from 0.5 to 1.0 SD, yet the variance is 0.087 when a standard deviation of 1.5 SD is used. In these cases, the reduced dataset contains the same number of model coefficients, and so the differences in variance are a direct result of differences in the perturbed input data, rather than the number of model coefficients stored.

For each value of  $\alpha > 0.1$ , increasing the standard deviation of the noise distribution decreases the variance in the reconstructed dataset in most cases. However, in other cases increasing the standard deviation of noise increases the variance in the data. When the variance of the reconstructed data increases, this is attributable to an increased number of model coefficients stored in the reduced dataset, although the variance of the reconstructed data only increases in 13 of 80 cases. Thus, in almost all cases, increasing the amount of noise in the data increases the variance in the reconstructed data when  $\alpha = 0.1$  (indicating a strong preference for reduction in error). However, when  $\alpha > 0.1$  (indicating a stronger preference for reduction in storage), increasing the amount of noise in the data leads to a decrease in the number of model coefficients stored and variance in the reconstructed data.

In contrast to the effects of noise, the removal of instances both at random and not at random does not substantially change the variance of the data. Though the variance is altered by a very small amount, the absolute change in variance is less than 1% for the traffic and air temperature datasets, and less than 2.3% for the rainfall dataset. Moreover, perturbing the data in space and time does not change the variance of the data. For both types of error, the variance in the reconstructed data decreases as  $\alpha$  increases and fewer model coefficients are stored.

### **Effect on Number of Partitions**

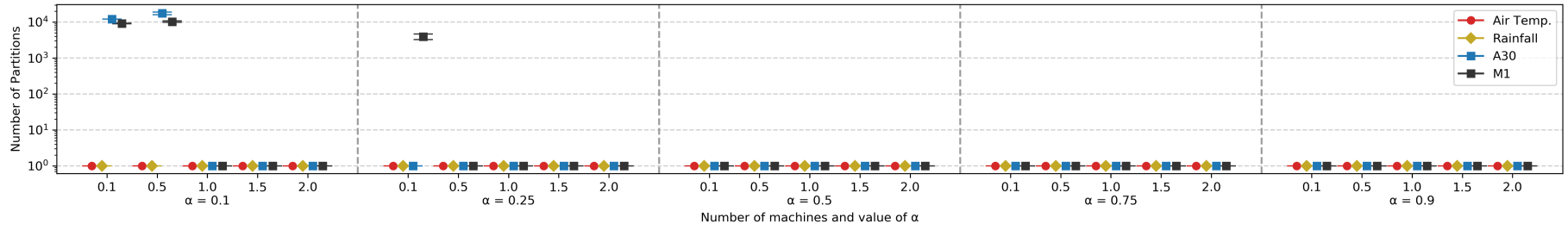
When  $\alpha = 0.1$ , increasing the amount of noise in the data increases the number of partitions when more than 1 cluster is used to partition the data. That is, when the reduced dataset contains more than 1 partition, increasing the standard deviation of noise in the features increases the number of partitions further. However, in many cases increasing the standard deviation results

in too much storage being used by the increased number of partitions. As a result, 1 fewer clusters are used in the partitioning process and the number of partitions output decreases. This can be seen in Figure 6.11a, which shows the number of partitions output when the four datasets are reduced using DTR-P modelling (results for the remaining modelling techniques can be seen in Appendix H.2). For example, increasing the standard deviation of noise from 0.5 to 1.0 SD decreases the mean number of partitions output from 17,523 to 1 for the A30 traffic dataset. Yet, as previously discussed, this does not affect the variance in the reconstructed dataset which continues to increase as the amount of noise in the input dataset increases.

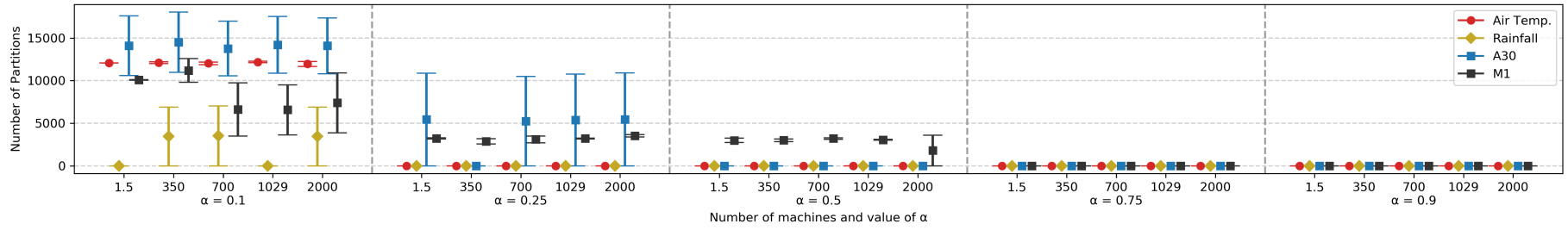
When  $\alpha > 0.1$ , the number of partitions output in almost all cases is 1. In cases where the number of partitions is greater than 1, increasing the standard deviation of noise decreases the number of partitions output to 1.

In some cases, perturbing instances in space and time leads to an increase in the number of partitions output in some cases. For example, when  $\alpha = 0.25$  and the M1 dataset is reduced using DCT-P modelling, the mean number of partitions increases from 3056 when a standard deviation of 1.5 m is used, to 3829 when a standard deviation of 2000 m is used. However, similar to the effects of noise, the number of clusters used for partitioning decreases as the amount of perturbation increases in these cases. Thus, the number of partitions decreases between 2 standard deviation values. In other cases, the number of partitions does not appear to change as the amount of perturbation increases, as shown in Figure 6.11.

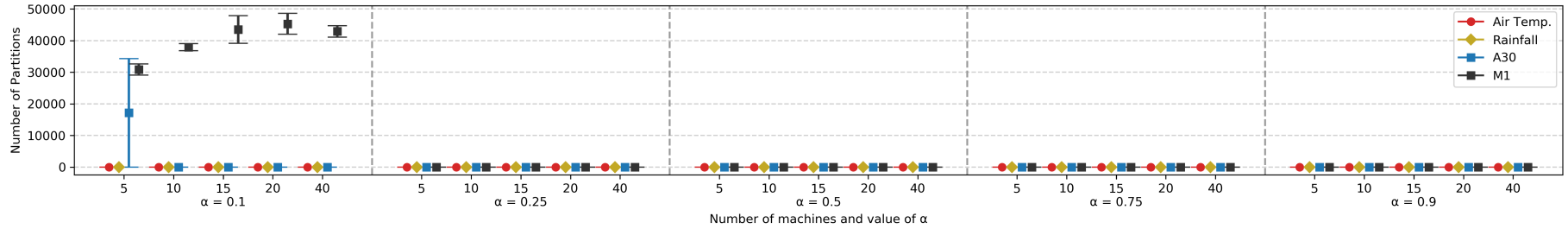
Increasing the number of missing instances from each dataset has a significant effect on the number of partitions output when  $\alpha \in \{0.1, 0.25\}$ . Figure 6.12 shows the effect of increased quantities of missing instances from the datasets, with results shown for DTR-P modelling, although similar results are observed for all modelling techniques (Appendix H.2). For example, when DCT-P modelling is used, increasing the amount of rainfall data that is missing at random leads to a decrease in the number of partitions output. However, in all other cases, a general trend is not exhibited when data is missing at random and missing not at random. Therefore, we may conclude that missing data can have a dramatic effect on the number of partitions output for values of  $\alpha$  close to 0, however the precise impact is not predictable. For values of  $\alpha$  close to 1, the number of partitions output is always 1, and so no change in partitions output is observed.



(a) Noise



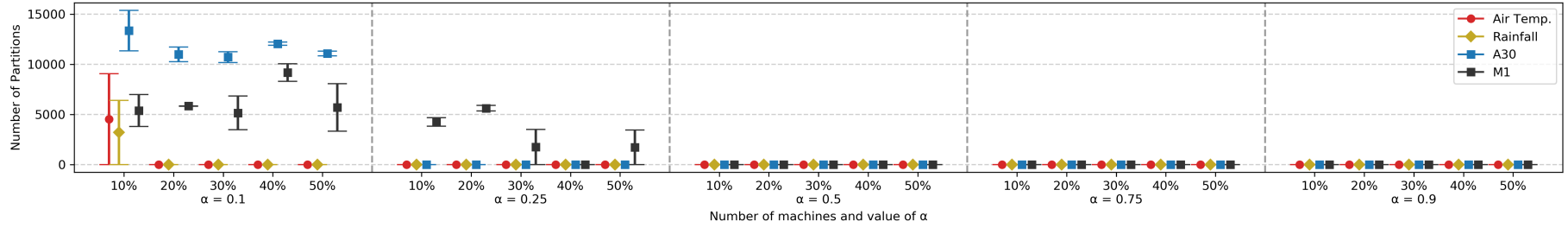
(b) Error in Space



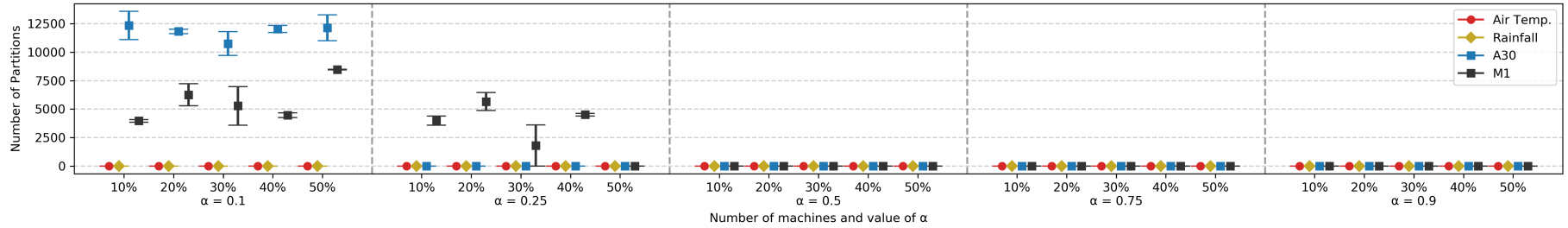
(c) Error in Time

Figure 6.11: Number of partitions in the reduced form of the datasets with noise and spatial/temporal error. Results are shown for DTR-P modelling. Note that in many cases, particularly when  $\alpha > 0.25$ , the number of partitions output is 1.





(a) Missing at Random



(b) Missing Not at Random

Figure 6.12: Number of partitions in the reduced form of the datasets with increasing amounts of missing instances. Results are shown for DTR-P modelling. Note that in many cases, particularly when  $\alpha > 0.25$ , the number of partitions output is 1.

### Effect on Reconstruction Error and Storage

The addition of noise increases the reconstruction error (NRMSE) of the reduced datasets in all cases (results figures can be seen in Appendix H.3). In most cases, increasing the amount of noise in the data increases the reconstruction error of the perturbed data. These cases are correlated with a similar or reduced volume of storage used as maintaining the models captures less of the variance in the perturbed data. For example, when  $\alpha = 0.1$ , the storage used by the air temperature dataset decreased from 3.2 MB to 1.6 MB when the amount of noise added increased from 0.1 SD to 2.0 SD. However, for the M1 dataset, the reconstruction error often decreases as the amount of noise in the input dataset increases, and in such cases the volume of storage used also increases. In contrast to the other datasets, this dataset contains many instances that are close in space but have significantly different values due to the presence of road works when the data is collected. Thus, the addition of noise reduces the difference between near instances that have large differences in feature values, allowing the models to better capture the variance of the data over space.

As the rate of missing data increases for the traffic and air temperature datasets, the reconstruction error and storage incurred is found to be directly correlated with the number of partitions in the reduced dataset. Moreover, when more than 1 partition is stored, the percentage storage used (relative to the input dataset volume) increases as the amount of missing data increases while the reconstruction error remains approximately the same. When only 1 partition is stored, the percentage storage used remains approximately the same while the reconstruction error increases. Note here that the percentage storage used indicates a less efficient reduction with respect to the input dataset, however the overall quantity of storage may still decrease as the volume of the input data also decreases.

However, for the rainfall dataset, a significantly increased number of model coefficients is stored when  $\alpha = 0.1$  compared to  $\alpha > 0.1$ . This contrasts to the other datasets, for which more partitions are stored when  $\alpha = 0.1$ . However, for the rainfall dataset when  $\alpha = 0.1$ , the error incurred does not increase as the amount of missing data increases, and the storage used remains approximately the same. Thus, we may infer that for normally distributed data, the rate of storage output may increase as the amount of missing data increases and the same number of model coefficients are used to store the reduced dataset. However, for data that has a long-tail distribution, the percentage storage used in the reduced dataset may decrease as the amount of missing data increases.

For all four datasets, the amount of storage used and reconstruction error incurred do not appear to change as the amount of spatial perturbation increases.

Moreover, the storage used and reconstruction error incurred remain correlated with the number of partitions output and appear unaffected by spatial and temporal perturbations. The only exception to this result is the A30 dataset when DTR-P modelling is used and  $\alpha > 0.5$ . Here, the reconstruction error increases as the amount of perturbation increases, while the number of model coefficients output remains the same.

### Effect on Imputation Error

Increasing the amount of noise in the data increases the imputation error for all three datasets when  $\alpha = 0.1$ , although the effect is less noticeable for the rainfall dataset. For example, when DTR-P modelling is used, increasing the standard deviation of noise from 0.1 to 2.0 SD increases the imputation error from 10.4% to 23.8% for the A30 traffic dataset, as shown in Figure 6.13 (results for all modelling methods and perturbations can be seen in Appendix H.4). However, when  $\alpha > 0.1$ , increasing the amount of noise has little effect on the imputation error. The only exception to this is an increase in error when the standard deviation increases from 0.1 to 0.5 for the M1 dataset, and DTR-P modelling is used with  $\alpha = 0.25$ . Here, the increase in imputation error is attributed to a decrease in the number of partitions stored.

For the traffic datasets when  $\alpha = 0.1$ , the imputation error incurred increases as the quantity of missing data increases. The same correlation also holds when  $\alpha = 0.25$  for the M1 dataset. This is attributed to the nearest sensors to the withheld instances being removed or placed in different partitions, thereby making the models used to impute the withheld instances less accurate at those locations. Yet, when  $\alpha > 0.1$  for the A30 dataset and  $\alpha > 0.25$  for the M1 dataset, increasing the quantity of missing data has little effect on the imputation error. In these cases few model coefficients are stored and so the removal of many instances has little effect on the models created. Furthermore, the removal of instances has little effect on the air temperature and rainfall datasets for all values of  $\alpha$ . The different results incurred for the traffic datasets versus the air temperature and rainfall datasets are attributable to the presence of significant discontinuities in space and time for the traffic datasets, and much smaller or no discontinuities for the air temperature and rainfall datasets. Note that removing instances both at random and not at random has similar effects on the imputation error.

Finally, perturbing the data in space has little effect on the imputation error incurred for the A30, air temperature and rainfall datasets. Like the presence of missing data, this lack of impact is attributable to similar instances being recorded at the same time in space. Thus, perturbing the data with a standard deviation of 2000 m has little impact as the models output are very

similar. However, for the M1 dataset, the imputation error incurred increases as the amount of perturbation increases when  $\alpha < 0.75$ . Like the impact of missing data on imputation error, this result is attributable to the significant difference between sensors that are near to each other in space in the M1 dataset. Perturbing these instances results in the models changing significantly at the locations of withheld instances, thereby increasing the error incurred when these withheld instances are imputed.

### Effect on Running Time

The addition of noise and spatial perturbations causes a minimal increase in the running time of the  $k$ D-STR algorithm. For example, when  $\alpha = 0.1$ , the mean running time of  $k$ D-STR on the air temperature dataset is 600 minutes when 0.1 SDs of noise are added, and 566 minutes when 2.0 SDs are added. However, a more noticeable increase is observed in some cases the data is temporally perturbed. Here, the mean running time is 4,573 minutes when a standard deviation of 5 minutes is used and 13,210 minutes when a standard deviation of 40 minutes is used. Conversely, removing data causes a decrease in running time. For example, when  $\alpha = 0.1$ , the mean running time of  $k$ D-STR on the air temperature dataset decreases from 6817 minutes to 2630 minutes when 50% of instances are removed. In all cases, the mean running time is correlated with the number of partitions output, with each reduction requiring a higher running time when more partitions are output.

### 6.3.4 Discussion

The addition of noise to a dataset increases the variance of the data. When  $\alpha = 0.1$ , the variance of the reconstructed data also increases as the standard deviation of noise in the input dataset increases. However, the variance of the reconstructed data is significantly lower than the variance of the input dataset, and the NRMSE of the reconstructed dataset increases as the standard deviation of noise increases, suggesting the models output are less able to capture the variance as the noise increases. Conversely, when  $\alpha > 0.1$ , increasing the standard deviation of noise leads to a decrease in variance in the reconstructed data. This is attributable to the same number or fewer model coefficients being stored, and so the NRMSE incurred increases both for reconstruction and imputation.

As the amount of missing data in each dataset increases, the variance in the remaining data varies but remains approximately the same. For the traffic datasets when  $\alpha = 0.1$ , and the M1 dataset when  $\alpha = 0.25$ , the number of partitions in the reduced dataset changes significantly as the amount of missing data increases. Accordingly, the amount of storage used varies and

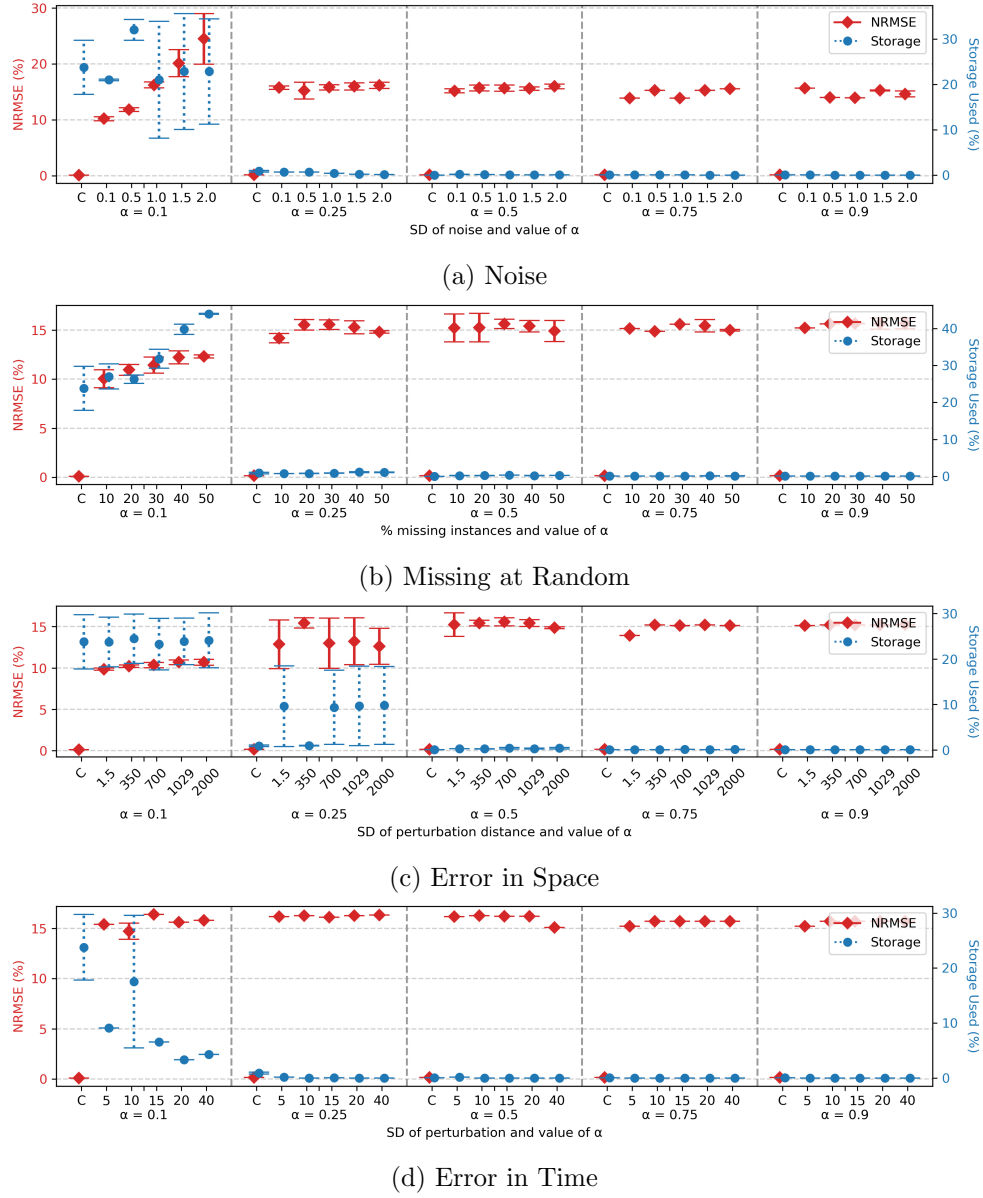


Figure 6.13: Imputation error incurred and storage used by erroneous A30 dataset. Results are shown for DTR-P modelling, and similar results are observed for MAR and MNAR.

the reconstruction and imputation errors vary too, although the same inverse correlation between storage used and incurred error shown in Chapter 4 is maintained. Yet, increasing the amount of missing data has little effect when  $\alpha > 0.25$ , and for all values of  $\alpha$  for the air temperature and rainfall datasets. In particular, this is attributable to only 1 partition and model being output in these circumstances. Furthermore, there appears to be little difference in output between data missing at random and missing not at random.

For the A30 traffic dataset, perturbing the data in the spatial domain leads to a small increase in reconstruction and imputation error, and the amount of error increases as the standard deviation of perturbation increases. For the M1 dataset, which has a lower distance between sensors and larger difference in feature values between adjacent instances, increasing the amount of spatial perturbation leads to a small increase in reconstruction error when  $\alpha < 0.5$ . This is correlated with fewer partitions being output and a lower volume of storage being used. However, increasing the amount of perturbation leads to significant increases in imputation error when  $\alpha < 0.75$ , and this is attributed to the large difference in feature values between adjacent instances.

Overall, these results only partially support Hypotheses H5.1 and H5.3. For H5.1, the amount of variance in the data increases over space and time but the number of partitions only increases in a limited number of cases. Moreover, the number of partitions actually decreases in some cases as the quantity of perturbation increases, and this is attributed to the increased storage overhead of the partitions and models, countered by the reduced accuracy of the models. For H5.3, increasing the amount of noise leads to an increase in imputation error for the traffic datasets when  $\alpha = 0.1$ , and has little to no effect when  $\alpha > 0.1$  and for the air temperature and rainfall datasets. The effect on imputation error of spatially or temporally perturbing the M1 dataset is only significant when  $\alpha < 0.75$ , and this is attributed to the large difference in feature values between adjacent instances. Furthermore, since the mean distance between sensors is much lower for the M1 dataset, perturbing the data in space has a much greater effect on the spatial distribution of instances compared to the A30, air temperature and rainfall datasets.

In contrast, Hypothesis H5.2 is fully supported by these results. In all cases, the modelling process reduces the impact of the perturbations, although in some cases the variance of the reconstructed data is still higher than the input dataset. An unexpected result is found — as the amount of noise increases, the variance in the reconstructed dataset decreases in some cases. However, this is attributable to fewer model coefficients being stored in those cases, likely due to the reconstruction error being too great thus forcing fewer model coefficients to be stored.

The results in this chapter offer no support for Hypothesis H5.4. As the

amount of missing data increases, the number of partitions output does not decrease. This is attributed to instances being removed from within partitions and those partitions still being retained by the remaining data, rather than those partitions being removed along with the missing instances. Hypothesis H5.5 is partially supported by the results in this section. Removing instances has little effect on the imputation error incurred for the air temperature and rainfall datasets, and again this is attributed to the partitions the instances were removed from still being created by the remaining instances in the input dataset. However, the imputation error increases considerably as data is removed from the A30 and M1 datasets for values of  $\alpha$  close to 0, and this is likely because of the high difference in feature values between adjacent instances. Removing the instances closest to the withheld instances causes the partitions and models used for imputation to change, thereby becoming less accurate for imputation.

## 6.4 Summary

In this chapter, alternative forms of data reduction for spatio-temporal data, and an analysis of  $k$ D-STR, were presented that address three significant challenges with ‘big data’: volume, heterogeneity and veracity. First,  $Mk$ D-STR was presented for mutually reducing multiple datasets, and shown to yield a more efficient reduction of multiple datasets while supporting mutual analysis tasks such as co-occurrence mining. This algorithm uses an alternative partitioning scheme to  $k$ D-STR and an adapted process that allows more model coefficients to be stored for more varied datasets while still allowing the user to control the overall reduction in storage and error with a single parameter. Second,  $Dk$ D-STR was presented for reducing datasets in a distributed computing environment. This adaptation allows the reduction process to scale as dataset sizes increase, and overcomes ambiguities on how partitioning should be performed across worker node boundaries. Finally, an analysis on the effects of noise, missing data and perturbations was presented. These results show that the data reduction process is effective at minimising the effects of noise in the data. However, as the rate of missing data and perturbations increases, the imputation error of the reduced dataset decreases as expected.

## Chapter 7

# Discussion

By reducing datasets to a series of partitions and models, the quantity of data used to store them and the time taken to process them is reduced. In this thesis, a novel method,  $k$ D-STR, for partitioning and reducing spatio-temporal data has been introduced. Furthermore, several adaptations of the algorithm have been proposed that overcome ambiguities and issues in a range of scenarios, and an analysis has been presented of the impacts of erroneous data on the algorithm.  $k$ D-STR and its variations meet the objectives outlined in Section 1.1, minimising both the storage used by the reduced dataset and the error incurred when reconstructing the data. Furthermore, analysis can be performed directly on the reduced dataset, and the reduced form can be used to impute instances and unsampled locations and times.

The findings of this thesis are examined throughout this chapter. First, Section 7.2 outlines how the contributions presented in Chapter 1 are met in Chapters 4, 5 and 6. Second, Section 7.2 discusses the utility of the reduction algorithms presented in this thesis. Finally, potential limitations on the generalisation of this research to other datasets and types of data are discussed in Section 7.3.

### 7.1 Contributions of this Research

The contributions of this thesis were outlined in Section 1.2 and are discussed here in further detail.

#### 7.1.1 Contribution 1: $k$ -Dimensional Spatio-Temporal Reduction

State-of-the-art reduction methods that partition and model a dataset present several shortcomings. In particular, they use fixed-size partitioning schemes that require the user to select an appropriate partition size, and few methods consider both the spatial and temporal autocorrelation in spatio-temporal data.



To overcome these issues, the  $k$ D-STR algorithm was proposed in Chapter 4. By using the variability of the data in its partitioning method,  $k$ D-STR overcomes the issues of fixed-size partitioning schemes. Furthermore, by using regression and DCT models,  $k$ D-STR creates models that allow data to be retrieved or imputed at any location and time within each partition. The algorithm also allows other modelling techniques that may be more appropriate or efficient for a particular dataset to be used. Thus,  $k$ D-STR meets the objectives defined in Chapter 1, overcoming the shortfalls of state-of-the-art algorithms while using a lower volume of data and incurring lower error in some cases.

In the analysis presented in Chapter 4, the parameter  $\alpha$  was shown to be effective at controlling the reduction in storage achieved and minimising the incurred reconstruction error. Results were presented for three datasets that exhibit different distributions in the feature space and spatio-temporal domain, as well as three different modelling techniques. These results show the generality of  $k$ D-STR for datasets commonly used within the transportation domain, and suggest similar results may be achieved for other sources of data that exhibit similar characteristics. Furthermore, the impact of increasing the quantity of data present while retaining the density of instances in space and time was shown, and a more efficient reduction may be achieved when a lower-dimensional SRS can be used.

### 7.1.2 Contribution 2: Augmenting and Reducing Spatio-Temporal Datasets

When augmenting a primary dataset with one or more supplementary datasets, the time taken to link the datasets can limit the volume of data that can be linked. In this context,  $k$ D-STR can be used to reduce the supplementary datasets, thereby speeding up the linking process as less data must be processed (as demonstrated in Section 5.5). Furthermore, the linked output dataset may be reduced, and a discussion on how to reduce and augment the datasets simultaneously is presented in Chapter 5.

However, the aim of the RES heuristic presented in Chapter 4 for  $k$ D-STR is to minimise the error incurred when reconstructing the original dataset. When reducing supplementary datasets, it is more useful to minimise the error in the features engineered during the linking process. Therefore, the LES heuristic was presented in Chapter 5 for this task, and results show this alternative heuristic incurred a lower NRMSE for linking tasks while using less storage. Moreover, the NRMSE incurred by the alternative heuristic was consistently lower or equal to the NRMSE incurred by the original heuristic given approximately the same storage used. These results show the alternative heuristic concentrates

model accuracy in those locations and times most relevant to the primary dataset. For common analysis tasks that link multiple datasets, the results presented in Chapter 5 demonstrate the benefits of reducing the supplementary datasets using the alternative heuristic.

### 7.1.3 Contribution 3: Mutually Reducing Datasets

When multiple datasets are analysed together, and they do not have an inherent primary and supplementary relationship, they can be reduced to a single set of common partitions. For each partition, a model is created and stored per dataset. This results in a single set of partitions that do not overlap, benefiting tasks such as co-occurrence mining wherein minimising the overlap between partitions can speed up the data mining process. However, the feature spaces of the datasets may not be comparable, making it ambiguous how the datasets should be clustered and partitioned together. That is, the datasets cannot be clustered together, presenting an issue for the partitioning process introduced in Contribution 1 (Chapter 4).

In Section 6.1, an adapted reduction algorithm was presented that overcomes this issue. This algorithm, Mutual  $k$ D-STR ( $Mk$ D-STR), makes use of an alternative partitioning process for multiple datasets that cover approximately the same area of space and period of time. Furthermore,  $Mk$ D-STR outputs more models and model coefficients for datasets with higher variation over space and time, thereby focusing information retention on those datasets. Empirical analysis shows  $Mk$ D-STR outputs fewer partitions than reducing the datasets independently for values of  $\alpha$  close to 0 and close to 1, resulting in a more efficient reduction overall.

### 7.1.4 Contribution 4: Distributed Data Reduction

To overcome the limits of reducing data on a single computer, a dataset can be spread across multiple compute nodes and reduced in a distributed fashion. Adapting parts of the  $k$ D-STR algorithm for distributed environments may be straightforward, such as using a distributed hierarchical clustering algorithm. However, it is ambiguous how partitions should be allowed to span the boundaries between compute nodes without incurring significant communication costs when modelling each partition. That is, allowing partitions to span boundaries results in data needing to be communicated between nodes during modelling, which impacts the speed of reduction. Yet, preventing partitions from spanning boundaries can incur increased imputation error.

To overcome this issue, Section 6.2 introduced the Distributed  $k$ D-STR ( $Dk$ D-STR) algorithm. This algorithm allows partitions to cross the boundaries between compute nodes, although restricts partitions to include data stored

on one compute node only. This removes the need for communication during modelling but retains accuracy during imputation. Furthermore, Section 6.2 shows how  $DkD$ -STR calculates the original  $kD$ -STR heuristic (Equation 4.8) in a distributed manner, and the alternative heuristic presented in Chapter 5 may be calculated in a similar manner. Empirical analysis in Section 6.2 shows that increasing the number of compute nodes results in a speedup of the reduction algorithm, allowing larger datasets to be reduced within a practical time frame, although the speedup achieved is correlated with the maximum number of instances on a compute node rather than the number of compute nodes. This analysis also shows that in some cases the imputation accuracy achieved is higher when partitions span compute node boundaries, although not all cases.

### 7.1.5 Contribution 5: Robustness to Error

When working with real-world datasets, the problems of noise, error in location and time, and missing data may affect the results achieved by algorithms for spatio-temporal data. To characterise the impact of these issues on  $kD$ -STR, an empirical analysis was presented in Section 6.3. In this analysis, varying degrees of these issues were introduced to four samples of data and the impact of these errors on the reduced and reconstructed datasets was measured.

These results show  $kD$ -STR reduces the impact of noise in the input, giving a lower variance in the reconstructed dataset. Furthermore, the amount of noise reduction can be implicitly controlled using the parameter  $\alpha$ , although the variance and reconstruction/imputation error of the data increases as the amount of noise increases in the input for values of  $\alpha$  close to 0. As the amount of missing data increases, the number of partitions in the reduced output, as well as the associated storage used and incurred NRMSE, can vary significantly for some datasets and values of  $\alpha$  close to 0. In empirical testing, this was found to be true for the traffic dataset, and this was attributed to the large differences in feature values between some instances that are close together in space. Removing some instances in those scenarios caused large differences in the incurred reconstruction error of the remaining instances, yielding the results given. For the other datasets, the storage used and incurred NRMSE of the reduced datasets did not change as the number of missing instances increased.

Similar to the effect of increasing the ratio of missing instances, increasing the amount of error in the spatial and temporal locations of instances caused small increases in reconstruction error for all three datasets and, in some cases, small increases in imputation error also. However, for the M1 dataset, the imputation error increased significantly as the amount of spatial perturbation

increased and  $\alpha < 0.75$ . Again this was attributed to the high differences in feature values of nearby instances for the M1 dataset (which is normally distributed), although the same result was not found for the rainfall dataset which also has significant differences between nearby instances but fewer occurrences (and exhibits a long-tail distribution).

## 7.2 Utility of Spatio-Temporal Reduction

Overall, the results presented in this thesis demonstrate the utility of reducing a range of spatio-temporal datasets using  $k$ D-STR and the variants presented in Chapters 5 and 6. Results have been shown for datasets that exhibit both normally and long-tail distributions in the feature space, and have varying amounts of spatio-temporal autocorrelation. In particular, the traffic datasets have high autocorrelation in some areas (e.g. long stretches of motorway) and low autocorrelation in others (e.g. at entries and exits), the air temperature dataset exhibits low autocorrelation in many areas, and the rainfall dataset exhibits high autocorrelation in space but low autocorrelation in time. Furthermore, by perturbing these datasets, the utility of  $k$ D-STR for noisy and erroneous datasets as well as datasets with high degrees of missing instances. For each of the datasets used for evaluation, the parameter  $\alpha$  was shown to be effective at controlling the storage used by the reduced dataset and the incurred error. For example, for a sample of the air temperature dataset, the storage used was shown to reduce to 22.9% of the original volume when  $\alpha = 0.1$  and 0.01% when  $\alpha = 0.9$ . For those values, the incurred reconstruction NRMSE was 3.5% and 10.7% respectively.

When more than 1 compute node can be used to reduce the data, the  $Dk$ D-STR adaptation of  $k$ D-STR has been shown to reduce the time required for reduction. This allows larger datasets to be reduced practically, where reducing them with a single compute node may have taken too long to be practical. When multiple datasets may be reduced together, the  $Mk$ D-STR algorithm has been shown to use less storage than reducing the datasets independently. Furthermore, the partitioning of space and time is the same for each of the datasets when  $Mk$ D-STR is used, reducing the number of partition comparisons for tasks such as co-occurrence mining. For other linking tasks, the LES heuristic and reduction process presented in Chapter 5 has been shown to yield more efficient and accurate reductions for such tasks. That is, when reducing supplementary datasets to speedup data augmenting tasks, the alternative heuristic is shown to reduce the NRMSE in the features engineered during linking.

The reduction algorithms presented in this thesis have been shown to be extensible for different modelling techniques, and the three techniques

evaluated yield different benefits for the user. Though only 1-dimensional and 2-dimensional SRSs have been evaluated, it is suggested the algorithms presented are applicable for any number of spatial dimensions. Thus, the algorithms presented throughout are extensible, robust to error, useful for datasets similar to those tested and may be useful for datasets exhibiting other characteristics also.

### 7.3 Limitations to Generalisation

The presence of autocorrelation in spatio-temporal data makes it ideal for modelling using few coefficients. However, when a dataset has low autocorrelation, the results achieved may be less useful than those presented in this thesis. Furthermore, several assumptions made about data and where it is sourced from are made in this thesis, and these may not always be applicable. Each of these assumptions present a limitation of the algorithms presented, and these are discussed in this section.

#### **Limitation 1** — Low autocorrelation

The presence of instances that are close in space and time that exhibit similar feature values is exploited by  $k$ D-STR both when partitioning and modelling the data. When the data has low autocorrelation, instances that are adjacent in space and time are likely to belong to different clusters. As a result, the number of partitions will be significantly higher in the reduced dataset. Furthermore, the instances within each partition may require a higher number of model coefficients to accurately capture them. Thus, data with very low autocorrelation may require more storage than the raw dataset while incurring a non-zero reconstruction error.

#### **Limitation 2** — Moving sensors

The partitioning process of  $k$ D-STR assumes each sensor is fixed in space does not move location, thereby allowing its location to be used as the centre of a Voronoi polygon. If sensors change location over time, either the polygon used to represent their location does not change or the new location is treated as a new sensor. Furthermore, as all sensors' polygons are used to partition the spatial domain of each time interval, the algorithm implicitly assumes few sensors will begin offline and become online after many time intervals, and that few will go offline before the end of the time period covered. When these assumptions are not met, no instance may be found at that time and location, and so the partitioning process may include these locations in other partitions incorrectly. As a result, the reconstruction of these instances may be significantly higher than expected, and so  $k$ D-STR should not be used in

this scenario.

**Limitation 3** — Multiple types of sensor at same location

A similar assumption made by  $k$ D-STR is that only 1 sensor may exist at a given location, and that each sensor may only record 1 instance at any given time. A further assumption made is that each instance has a value for each feature. However, in many datasets, multiple sensors may be located at the same location and different types of sensor may be used in the same dataset, and so each sensor may record some features but not others. As a result, the clustering technique used within the partitioning method of  $k$ D-STR may not be able to cluster these datasets, and so the datasets cannot be partitioned or reduced. However, other clustering techniques may be used that can overcome this limitation, thereby enabling  $k$ D-STR to reduce such datasets.

**Limitation 4** — Streaming data

A final assumption made by  $k$ D-STR is that all instances are present throughout the reduction process. That is, all instances are available to be clustered at the start of the reduction process, and may be compared and modelled as the process iterates. However, in many scenarios the data may only be processed in batches, or each instance must be processed as it arrives at a worker node and cannot be stored or retrieved after being processed. In these scenarios,  $k$ D-STR cannot be used to reduce the data, and so adaptations of the algorithm for streaming data are required.

## Chapter 8

# Conclusions and Future Work

This thesis has explored the problem of reducing the storage overhead of spatio-temporal data and making a dataset faster to process. Specifically, this research lies within the domain of reduction methods for spatio-temporal data science, and built upon the fields of clustering and data modelling. The research proposed and evaluated algorithms with the following objectives: (i) to minimise the storage used by the reduced dataset, (ii) to minimise the loss of information in space and time, and of the dataset’s features, (iii) to maximise the accuracy of the reduced dataset, and (iv) to permit analysis using the reduced form of the dataset. These objectives were maintained throughout the algorithms and adaptations presented in Chapters 4, 5 and 6.

In Chapter 4, a novel data reduction algorithm,  $k$ D-STR, was proposed and evaluated. This algorithm takes advantage of the spatio-temporal autocorrelation in the data to reduce the dataset to a smaller form, while minimising the storage used and error incurred. By varying a single parameter, the user is able to control the loss in storage versus the increase in incurred error. Through the analysis presented in Chapter 4,  $k$ D-STR was shown to be effective for datasets exhibiting a range of characteristics, and is extensible for different SRSs and volumes of data.

In Chapter 5, the  $k$ D-STR algorithm was integrated with the common analysis task of augmenting a spatio-temporal dataset with 1 or more supplementary datasets. The evaluation presented in Chapter 5 demonstrated the utility of reducing datasets prior to linking, speeding up the process of augmenting a primary dataset. However, when reducing a supplementary dataset prior to linking, the reconstruction error of the original features is less important than the error in the features engineered during linking. Therefore, Chapter 5 also introduced an alternative heuristic for reducing supplementary datasets that prioritises error reduction in the engineered features. This was shown to concentrate information retention in the areas of the supplementary dataset most applicable to the primary dataset, yielding a more efficient reduction that

is more accurate.

Chapter 6 analysed and adapted the  $k$ D-STR reduction algorithm for three properties of ‘big data’. First, the problem of heterogeneity was considered and the Mutual  $k$ D-STR (M $k$ D-STR) algorithm proposed for reducing multiple datasets simultaneously. This adapted algorithm uses an alternative partitioning process for multiple datasets and is able to reduce those datasets together. The algorithm concentrates information retention in the most varied dataset, and was shown to yield a more efficient reduction than reducing the datasets separately in many scenarios. Second, the problem of data volume was considered and the Distributed  $k$ D-STR (D $k$ D-STR) algorithm presented for reducing a dataset in a distributed fashion. This algorithm overcomes inherent ambiguities when distributing the reduction process over multiple compute nodes, removing the need for communication between the compute nodes when modelling the data. Empirical analysis demonstrated the speedup that can be achieved by reducing the data with this distributed algorithm, allowing significantly larger datasets to be reduced in reasonable time.

Finally, Chapter 6 analysed the impact of veracity or error in the input data on the reduction process. By perturbing the instances in the input datasets, adding noise to the feature values and removing instances, the impact of error, noise and missing data were investigated. This analysis is of practical benefit to users of the  $k$ D-STR reduction process, demonstrating the impact on results achieved by such errors. Furthermore, these results indicate what results may be achieved for datasets with similar characteristics to those used.

Throughout this thesis, all experiments utilised real-world spatio-temporal datasets commonly used within the transportation domain. These datasets, namely road traffic, air temperature and rainfall, exhibit a range of spatio-temporal variability and distributions, and demonstrate the results that may be achieved for similar datasets in other domains. These results show the benefit of reducing the data, both in reducing the storage overhead of the data and in speeding up common tasks such as data linking. Combined, the algorithms presented in this thesis and the results of the experiments undertaken offer both a novel theoretic approach for partitioning and reducing datasets, and practical algorithms that may be of benefit to data scientists.

## 8.1 Future Work

Multiple opportunities of future research have been created as a result of the research presented in Chapters 4, 5 and 6. In this thesis, Contribution 1 has shown that spatio-temporal data may be reduced to a significantly smaller form by taking advantage of the data’s autocorrelation when partitioning and modelling the data, and Contribution 2 has shown this approach is extensible



for specific scenarios such as linking datasets. Contributions 3 and 4 have shown that adaptations of  $k$ D-STR can be used for mutually reducing multiple datasets and reducing datasets in a distributed setting, while Contribution 5 has shown the robustness of  $k$ D-STR to noise, error and missing data. As such, the following suggestions provide a set of potential avenues for future research that extend the generality of these algorithms to other sources of data, and research that improves the efficiency and accuracy of the reduced output.

1. The algorithms presented in this thesis could be used to reduce other sources of data that exhibit different characteristics to those used. Experimentation with other modelling techniques may reveal alternatives that yield a more efficient reduction or one that incurs a lower error for datasets with different characteristics. Furthermore, we may investigate utilising the correlation between features when modelling the data, rather than assuming each feature to be independent of the others.
2. In addition, further heuristics may be investigated that are more useful for a given task or type of data. For example, it may be beneficial to maximise the variance between neighbouring partitions, where this is not guaranteed with  $k$ D-STR's partitioning technique. In scenarios where some sensors give less accurate recordings or have a lower utility than others, the utility of the instances within each partition and the subsequent confidence of the models they yield may be considered. This may yield a less efficient reduction but maximise the user's confidence in the accuracy of the models generated.
3. A third avenue is to investigate feature engineering during the modelling process. For example, while location and time may be useful at capturing the spatio-temporal variability of the data, engineered features such as day of week or urban/rural may offer more generalisable models that are more useful. These features can be engineered from location and time, and so instances can still be retrieved or imputed for any location and time.
4. Repeating patterns often feature in spatio-temporal datasets. For example, air temperature data follows daily patterns of increased and decreased values, and traffic data follows both daily and weekly patterns. The algorithms presented in this thesis do not take advantage of these repeating patterns and may treat the same period of time for 2 consecutive days as different partitions despite them containing similar instances. A more efficient reduction may be created if these patterns are considered.
5. When working with very large sources of data, data is often generated in streams that continuously produce new instances. In such cases, instances

must be processed as they are generated and it is infeasible to store more than a short history of data. However, the algorithms presented in this thesis require knowledge of the full dataset. Where data becomes available in batches,  $k$ D-STR may be used to reduce each batch independently. However, further research may yield a more useful reduction algorithm for streams of data, both when data may be processed in batches and when each instance must be processed immediately.

## Appendix A

# Supplementary Results for Chapter 4

The results in this appendix complement the discussion of results presented in Section 4.4.

### A.1 Trade-Off Between Error and Storage

Figure A.1 shows how the error incurred and storage used varies more precisely for the air temperature dataset when PLR-P modelling is used and  $0.1 \leq \alpha \leq 0.24$ . When  $\alpha = 0.1$ , the instances are clustered into 3 clusters resulting in 15,596 partitions. When  $\alpha = 0.12$ , the instances are clustered into 2 clusters, resulting in 11,494 partitions, and the instances are clustered into 1 cluster resulting in 1 partition when  $\alpha > 0.12$ .

Figure A.2 shows the number of coefficients output per model. When  $\alpha > 0.12$ , only 1 partition and 10 model coefficients are output, demonstrating that increasing the value of  $\alpha$  between 0.12 and 0.25 does not limit the number of coefficients output. However, when  $\alpha = 0.5$ , 4 coefficients were output, and only 1 coefficient was output when  $\alpha \in \{0.75, 0.9\}$ .

### A.2 Reconstructed Sensor Values

Figure A.3 shows the true values recorded for 5 sensors on the 1st June 2017. The reconstructed values after the datasets were reduced using PLR-P modelling with  $\alpha = 0.1$  and  $\alpha = 0.9$  are also shown. These figures demonstrate the improvement in reconstruction error that is achieved by reducing  $\alpha$  from values close to 1 to values close to 0. When  $\alpha = 0.1$ , more than 1 partition is stored for the traffic and air temperature datasets, allowing the models to more accurately capture the variance of the data. In comparison, when  $\alpha = 0.9$  for all datasets and when  $\alpha = 0.1$  for the rainfall dataset, the number

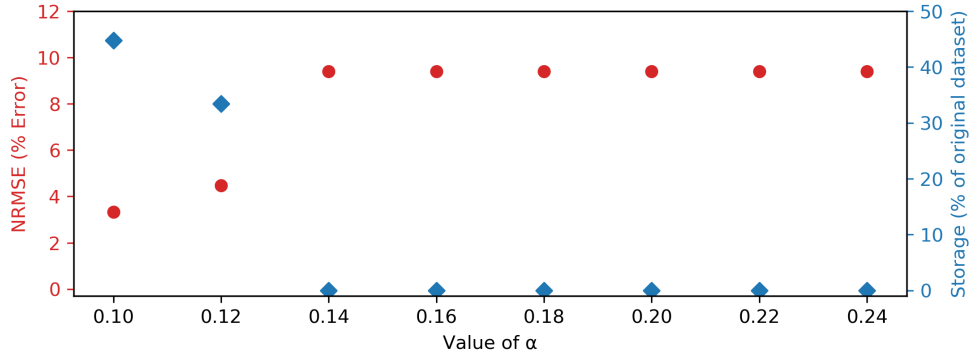


Figure A.1: Storage and error incurred for the air temperature dataset with PLR-P modelling.

of partitions stored is 1 and the single model captures the variance of the data less accurately.

These results also demonstrate the variation in reconstruction errors between sensors. For example,  $k$ D-STR is able to accurately capture the values for sensor 50 in the Air Temperature dataset when  $\alpha = 0.1$ , while less accurately capturing the values for sensor 67. At sensor 50, fewer partitions were stored than at sensor 67, though more coefficients were stored for one of those partitions. This allowed the models to more accurately capture the variance of sensor 50, while only a mean value was stored per model for sensor 67.

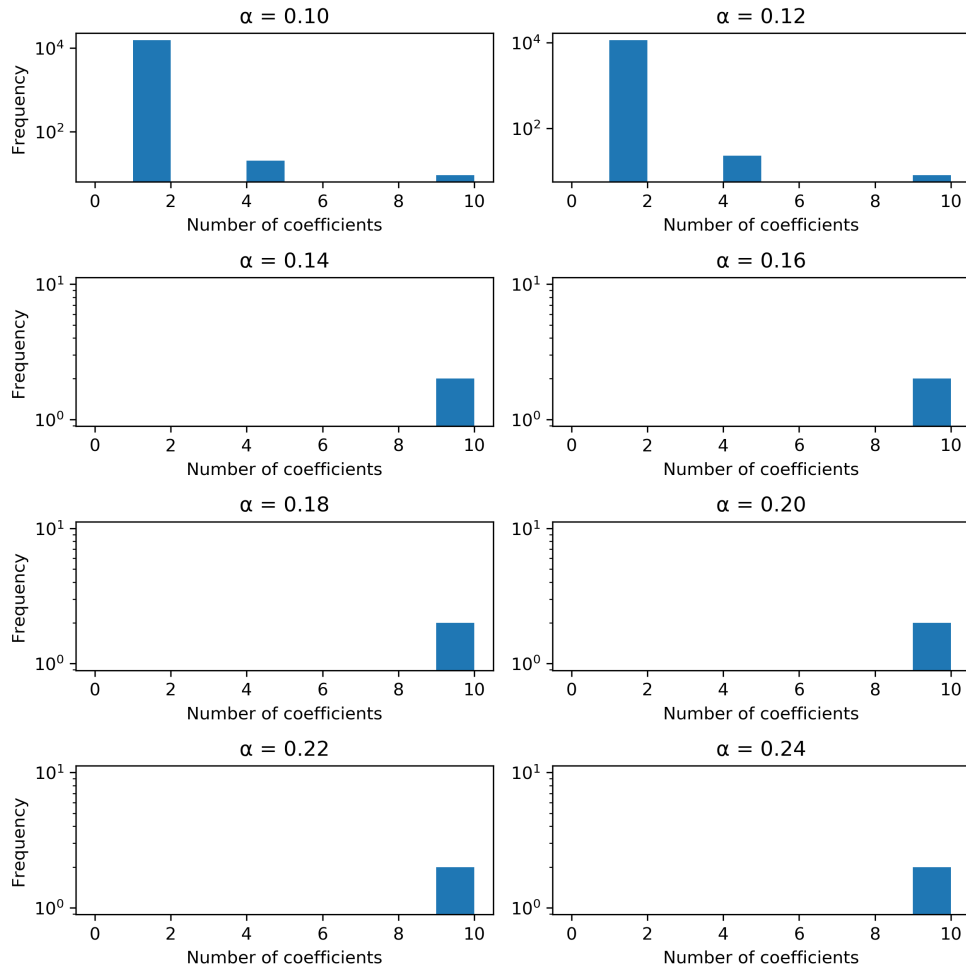
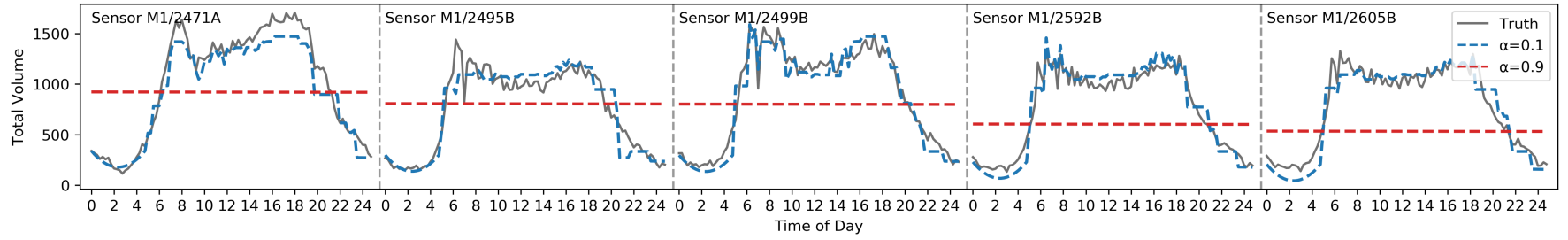
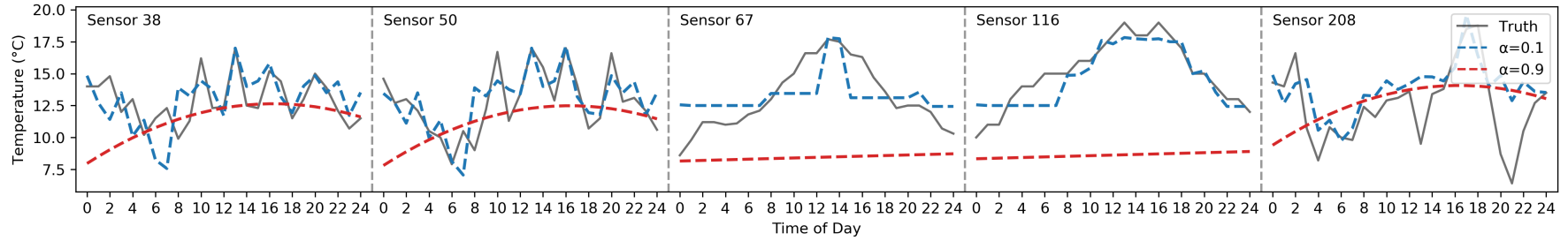


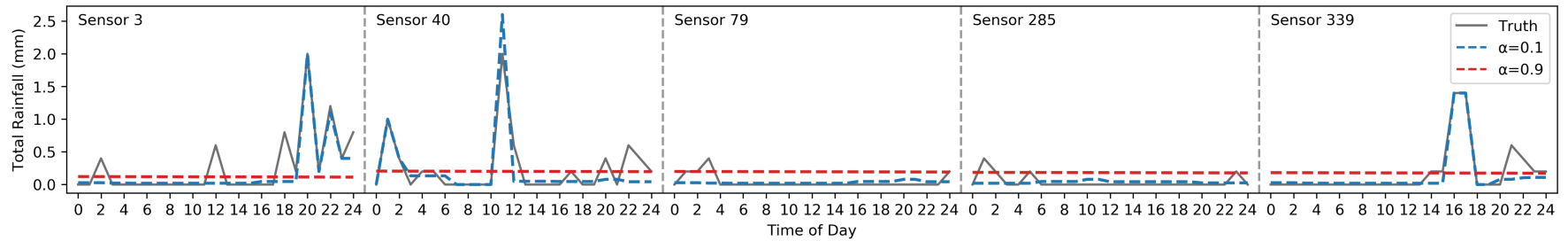
Figure A.2: Histogram of coefficients stored per partition for the air temperature dataset with PLR-P modelling.



(a) Traffic



(b) Air Temperature



(c) Rainfall

Figure A.3: NRMSE incurred and storage used by the reduced traffic, air temperature and rainfall datasets.

### A.3 Volumes of Partitions

These heatmaps show the volumes of partitions for the traffic and air temperature datasets when  $\alpha \in \{0.1, 0.25\}$ . For the traffic datasets, many partitions at nighttime include all sensors as indicated by many partitions containing many sensors for few time intervals. Note that figures are not shown for  $\alpha > 0.25$ , and for the rainfall dataset, as these reduced datasets contain 1 partition encompassing all sensors and time intervals.

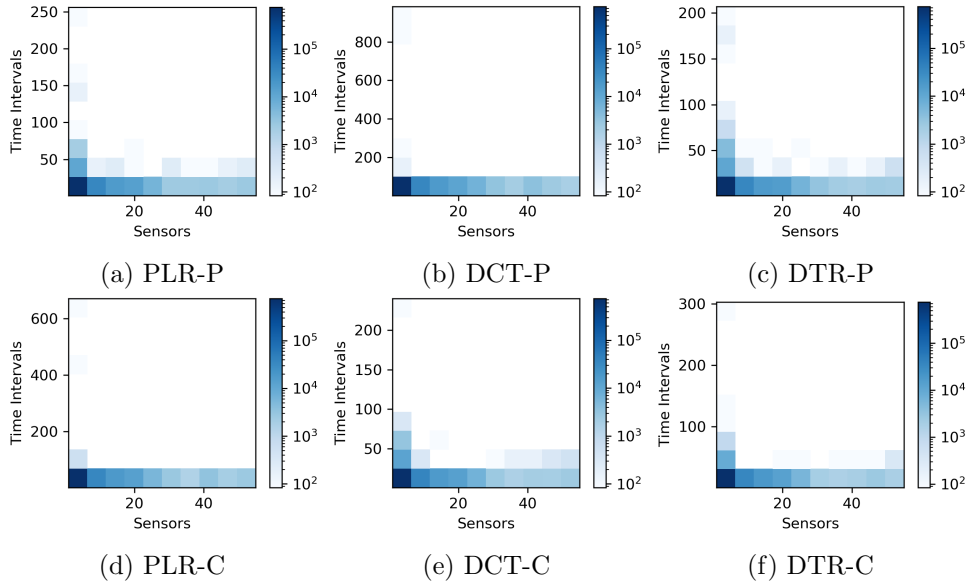


Figure A.4: Volumes of Partitions (traffic datasets when  $\alpha = 0.1$ ).

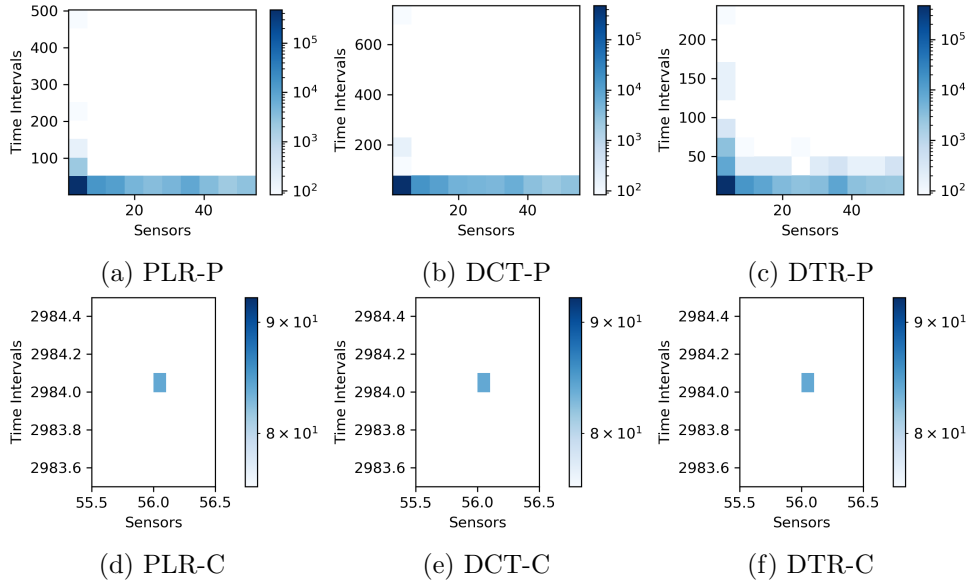


Figure A.5: Volumes of Partitions (traffic datasets when  $\alpha = 0.25$ ).

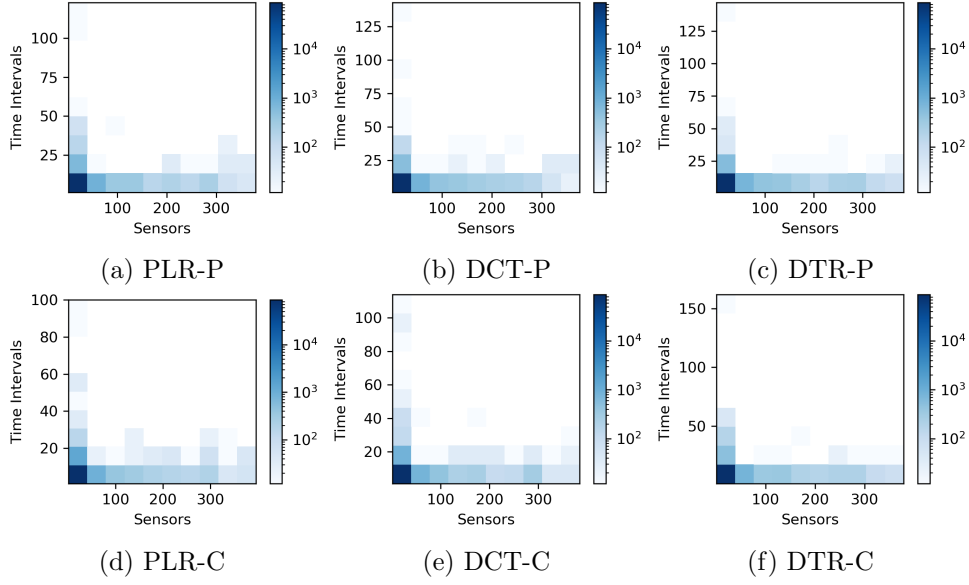


Figure A.6: Volumes of Partitions (air temperature dataset when  $\alpha = 0.1$ ).

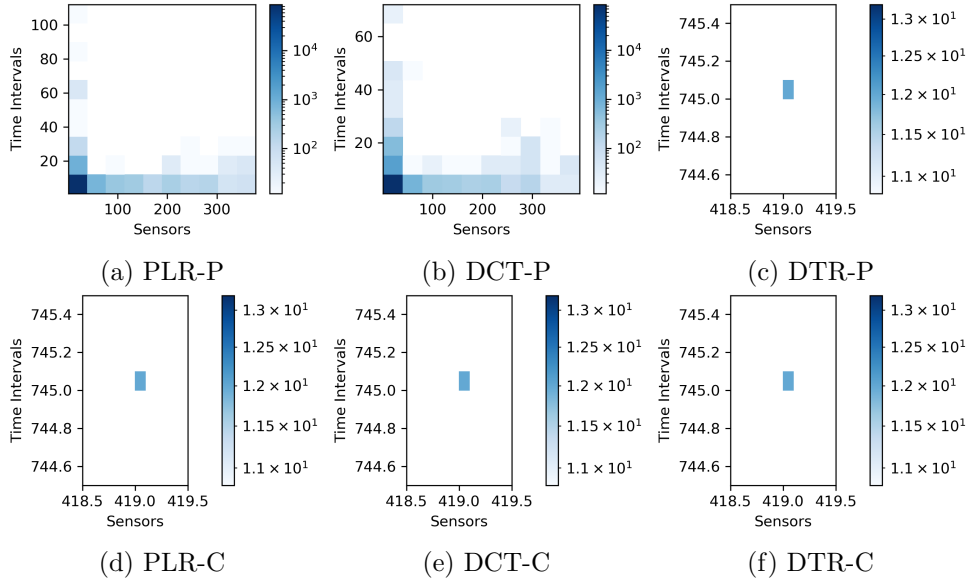


Figure A.7: Volumes of Partitions (air temperature dataset when  $\alpha = 0.25$ ).

## A.4 Effect of Distance Metric

Changing the distance metric used in the clustering step can change the number of partitions output, as shown in Table A.1. Results are shown for the M1 traffic dataset, when PLR-P modelling is used and  $\alpha = 0.1$ . However, while the number of partitions can vary, the relationship between the number of partitions, storage used and error incurred remains the same. This is shown in Figure A.8.



Table A.1: Number of partitions output for the M1 traffic dataset when  $\alpha = 0.1$  and PLR-P modelling is used.

Distance Metric	Partitions Output		
	Min	Mean	Max
$L_1$ (Manhattan)	5975	7857	10581
$L_2$ (Euclidean)	5561	7353	8941
$L_\infty$ (Chebyshev)	1918	5661	10838

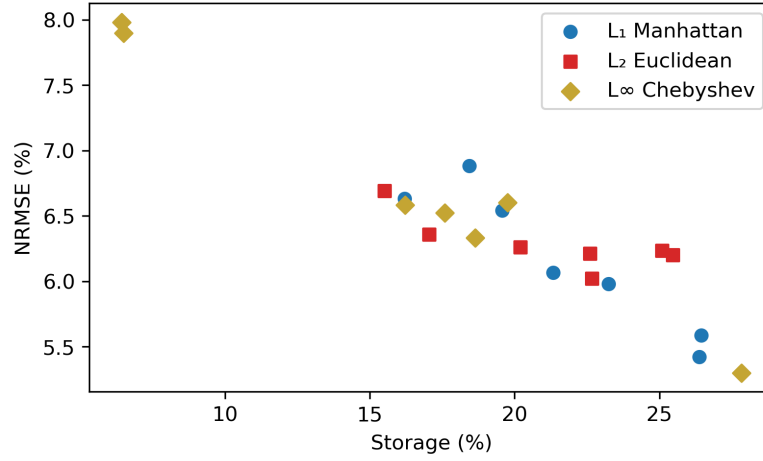


Figure A.8: Storage and error incurred for the M1 traffic dataset when PLR-P modelling is used and  $\alpha = 0.1$ .

## A.5 Choice of Spatial Referencing System

Table A.2: Partitions output when using a 1D versus 2D SRS for the traffic datasets. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

Techn.	$\alpha$	1D SRS			2D SRS		
		Min.	Mean	Max.	Min.	Mean	Max.
PLR-P	0.1	5918	11997	27485	1630	8694	27837
	0.25	2744	11993	27082	1	5182	18783
	0.5	2290	8167	23080	1	407	5422
	0.75	1	4415	12645	1	1	1
	0.9	1	7	254	1	1	1
PLR-C	0.1	3409	21666	43589	1479	7193	27878
	0.25	2145	15389	33108	1	2359	10309
	0.5	1	7391	24236	1	93	2337
	0.75	1	802	8406	1	1	1
	0.9	1	7	251	1	1	1
DCT-P	0.1	2152	8199	17937	1634	6468	13794
	0.25	1	7603	24689	1440	5076	13583
	0.5	1	544	8974	1	648	6649
	0.75	1	1	1	1	1	1
	0.9	1	1	1	1	1	1
DCT-C	0.1	1914	13519	38766	1467	8165	21041
	0.25	2000	12022	30518	1	1749	9142
	0.5	1	6688	19090	1	22	1753
	0.75	1	526	5363	1	1	1
	0.9	1	1	1	1	1	1
DTR-P	0.1	2737	12628	27452	1638	9084	27843
	0.25	2133	9590	24574	1	4971	13601
	0.5	1	5868	17978	1	297	3724
	0.75	1	53	3894	1	1	1
	0.9	1	4	217	1	1	1
DTR-C	0.1	5782	20237	45431	1465	7423	27776
	0.25	2146	13036	31732	1	2263	10288
	0.5	1	6089	24188	1	50	2361
	0.75	1	257	5007	1	1	1
	0.9	1	1	1	1	1	1

## A.6 Impact of Input Dataset Size

The figures presented in this section show the storage used by the reduced datasets and NRMSE incurred as the dataset size increases in the temporal and spatial domains. The figures presented show the impact on  $k$ D-STR with PLR modelling, though figures for the remaining modelling techniques are available in an online repository [120]. Furthermore, the tables presented show the number of partitions output as the dataset size increases.

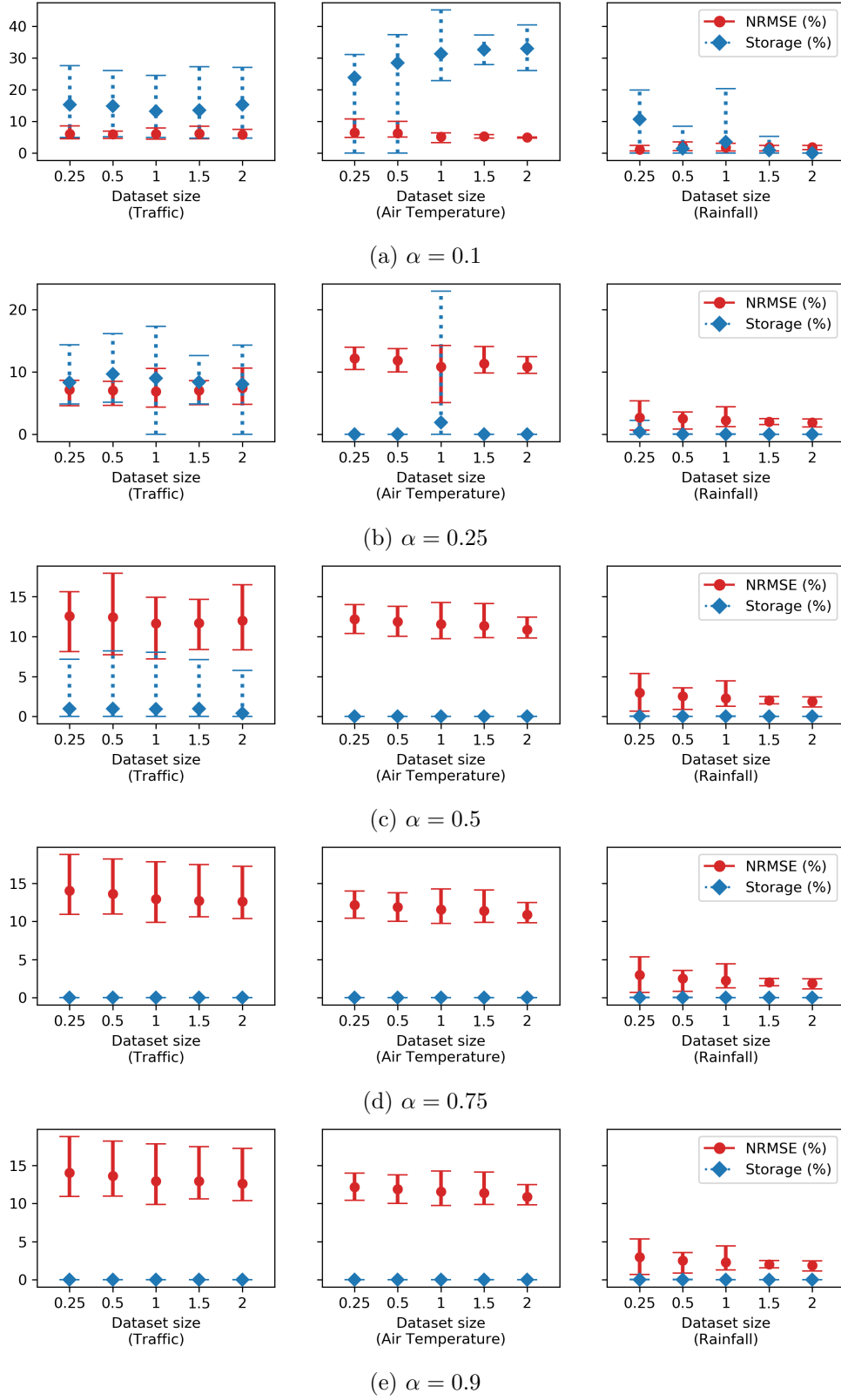


Figure A.9: Effect on NRMSE and storage used as the quantity of data in the temporal domain increases, with results shown for PLR modelling.

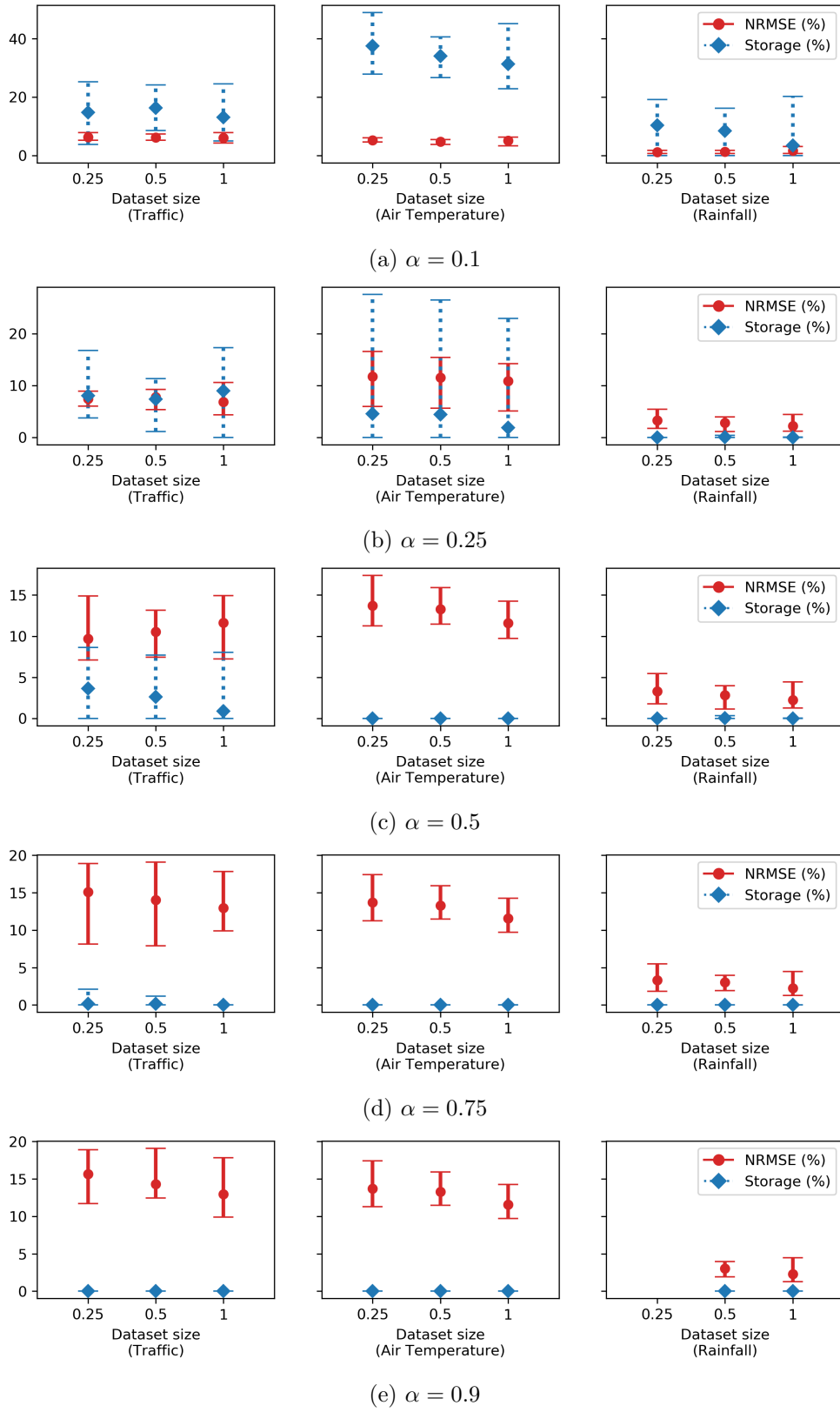


Figure A.10: Effect on NRMSE and storage used as the quantity of data in the spatial domain increases, with results shown for PLR modelling.

Table A.3: Average number of partitions stored as the quantity of data increases in the spatial and temporal domains for the traffic and air temperature datasets. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

		Traffic Datasets									Air Temperature Dataset							
Techn.	$\alpha$	Temporal Domain					Spatial Domain			Temporal Domain					Spatial Domain			
		0.25×	0.5×	1×	1.5×	2×	1×	1.5×	2×	0.25×	0.5×	1×	1.5×	2×	1×	1.5×	2×	
PLR	0.1	2875	5130	8429	12772	18971	2773	5145	8429	1899	4594	9921	15535	21144	3177	5540	9921	
	0.25	1316	3256	5772	7619	10057	1469	2465	5772	1	1	637	1	1	423	756	637	
	0.5	116	257	425	651	378	699	696	425	1	1	1	1	1	1	1	1	
	0.75	1	1	1	1	1	16	31	1	1	1	1	1	1	1	1	1	
	0.9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
DCT	0.1	1662	3676	7494	10344	12881	2736	4300	7494	2503	5569	9618	15468	21247	3190	5499	9618	
	0.25	1265	3013	5603	7515	10163	1461	2468	5603	1	812	2224	2478	2920	803	1599	2224	
	0.5	331	257	730	907	1715	542	696	730	1	1	1	1	1	1	1	1	
	0.75	1	1	1	1	1	15	31	1	1	1	1	1	1	1	1	1	
	0.9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
DTR	0.1	2668	5474	9184	15144	16357	2628	5485	9184	2440	5519	9598	15442	21237	2856	5460	9598	
	0.25	1270	3262	5471	7505	10204	1160	2343	5471	1	1	1	1	1	1	1	1	
	0.5	116	1	284	646	381	277	549	284	1	1	1	1	1	1	1	1	
	0.75	1	1	1	1	1	15	31	1	1	1	1	1	1	1	1	1	
	0.9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Table A.4: Average number of partitions stored as the quantity of data increases in the spatial and temporal domains for the rainfall dataset. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

Techn.	$\alpha$	Rainfall Dataset							
		Temporal Domain					Spatial Domain		
		0.25×	0.5×	1×	1.5×	2×	1×	1.5×	2×
PLR	0.1	885	232	1215	502	1	994	1527	1215
	0.25	34	2	3	1	1	1	14	3
	0.5	2	2	3	1	1	1	14	3
	0.75	2	2	2	1	1	1	1	2
	0.9	2	3	1	1	1	1	1	1
DCT	0.1	848	1188	2448	1632	5092	1078	1429	2448
	0.25	30	3	3	1	1	1	14	3
	0.5	30	2	3	1	1	1	13	3
	0.75	2	2	3	1	1	1	1	3
	0.9	2	3	1	1	1	1	1	1
DTR	0.1	420	241	1377	524	1	1001	1520	1377
	0.25	32	3	2	1	1	1	13	2
	0.5	2	2	2	1	1	1	14	2
	0.75	2	2	3	1	1	1	1	3
	0.9	2	2	3	1	1	1	1	3

## Appendix B

# Creating Baseline Augmented Datasets

To create a baseline augmented dataset using the raw datasets, the  $k$  nearest neighbours (kNN) and fixed-sized neighbourhood methods were used, as well as their inverse distance weighted (IDW) variants. For each primary instance, the daily air temperature, traffic volume and rainfall features were estimated using the instances recorded at the  $k$  nearest stations in each dataset, or all stations within a set radius of the section. The linking procedure shown in Algorithm 7 was used to augment the raw road condition dataset  $D^P$  with information from the three raw supplementary datasets  $D^1, D^2$  and  $D^3$ .

---

**Algorithm 7:** Augmenting the road condition dataset  $D^P$  with features calculated from the traffic, air temperature and rainfall incident on each section of road

---

```
1  $D^{\text{Aug}} \leftarrow \emptyset$  // Initialise the output linked dataset to empty
2 for  $d_{s,t} \in D^P$  do
3   if  $\text{lastInstance}(d_{s,t}, D^P)$  then
4      $d_{s,t'} \leftarrow \text{lastInstance}(d_{s,t}, D^P)$  // Get last instance at same
        location
5     for  $1 \leq l \leq 3$  do
6        $L^l \leftarrow \emptyset$  // All instances between  $t$  and  $t'$  from  $D^l$ 
7       for  $t \leq t_c \leq t'$  do
8          $L^l[t_c] \leftarrow \text{link}(D^l, s, t_c)$  // Impute instance for  $D^l$  for day
             $t_c$  at location  $s$ 
9       end
10       $d'_{s,t} \leftarrow d_{s,t} + \text{features}(d_{s,t}, l)$  // Add calculated features onto
           $d_{s,t}$ 
11    end
12     $D^{\text{Aug}} \leftarrow D^{\text{Aug}} + d'_{s,t}$ 
13  end
14 end
```

---

In Algorithm 7, line 2 iterates over all instances in  $D^P$ , the road condition dataset. For each instance, line 3 checks if there exists another instance in  $D^P$  at the same location on a day prior to the instance being considered. If there is such a previous instance, the date that instance was recorded is assigned to  $t'$ . Then, line 5 iterates over each supplementary dataset, that is the air temperature, rainfall and traffic datasets. Lines 6—9 impute all instances from supplementary dataset  $D^l$  between the dates  $t'$  and  $t$  at the location  $s$ , using the kNN technique or neighbourhood technique. Finally, line 10 augments the original instance  $d_{s,t}$  to create the linked feature  $d'_{s,t}$ , and line 12 adds the augmented instance to the output dataset. Since the location and time of each primary instance may be compared to every supplementary instance, this algorithm has a time complexity of  $\mathcal{O}(|D^P|^2 nx)$ . Here,  $|D^P|$  is the number of instances in the primary dataset,  $n$  is the number of supplementary datasets, and  $x$  is the maximum number of instances in any of the supplementary datasets.

To determine which of value of  $k$  and which neighbourhood size to use for each dataset, 10-fold cross validation was used and the withheld instances imputed using each of the techniques and a range of parameter values. This process was completed offline and followed the procedure used by domain experts who regularly link these datasets. The accuracy of these imputations can be seen in Figures B.1, B.2 and B.3.

These results show that a value of  $k = 5$  gives the lowest median error for the kNN and IDW kNN methods on the traffic and air temperature datasets. For the rainfall dataset, a value of  $k = 1$  gives the lowest median error when imputing rainfall data. When using the neighbourhood methods, a value of 0 miles and 0 days was found to give the lowest median error for all three datasets. However, this range was not sufficiently large to impute every withheld instance. Instead, a range of 20 miles and 0 days yielded the lowest median error for the traffic dataset, while a range of 40 miles and 0 days yielded the lowest median error for the air temperature and rainfall datasets while allowing every instance to be imputed.



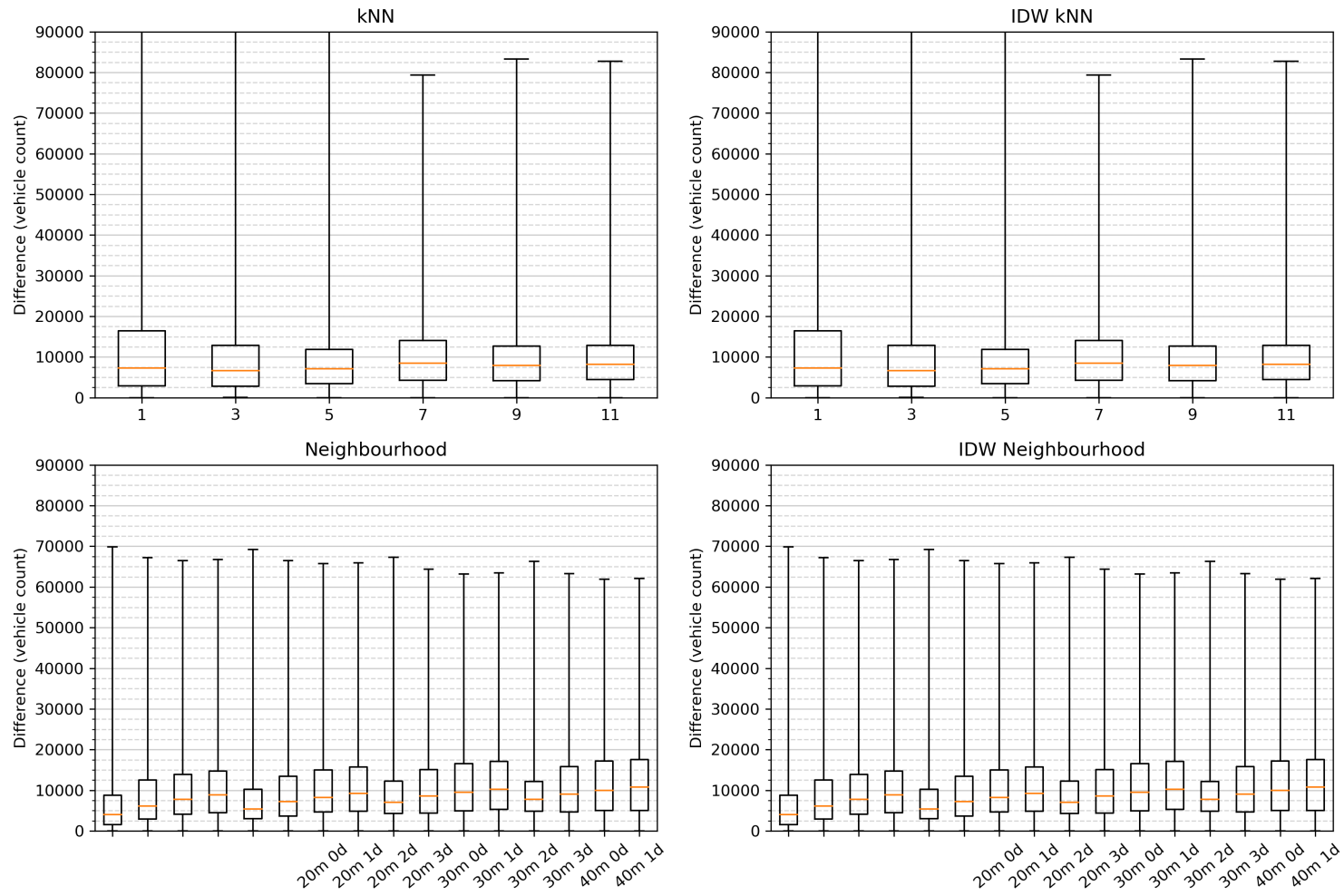


Figure B.1: Accuracy of imputing withheld instances for the traffic dataset

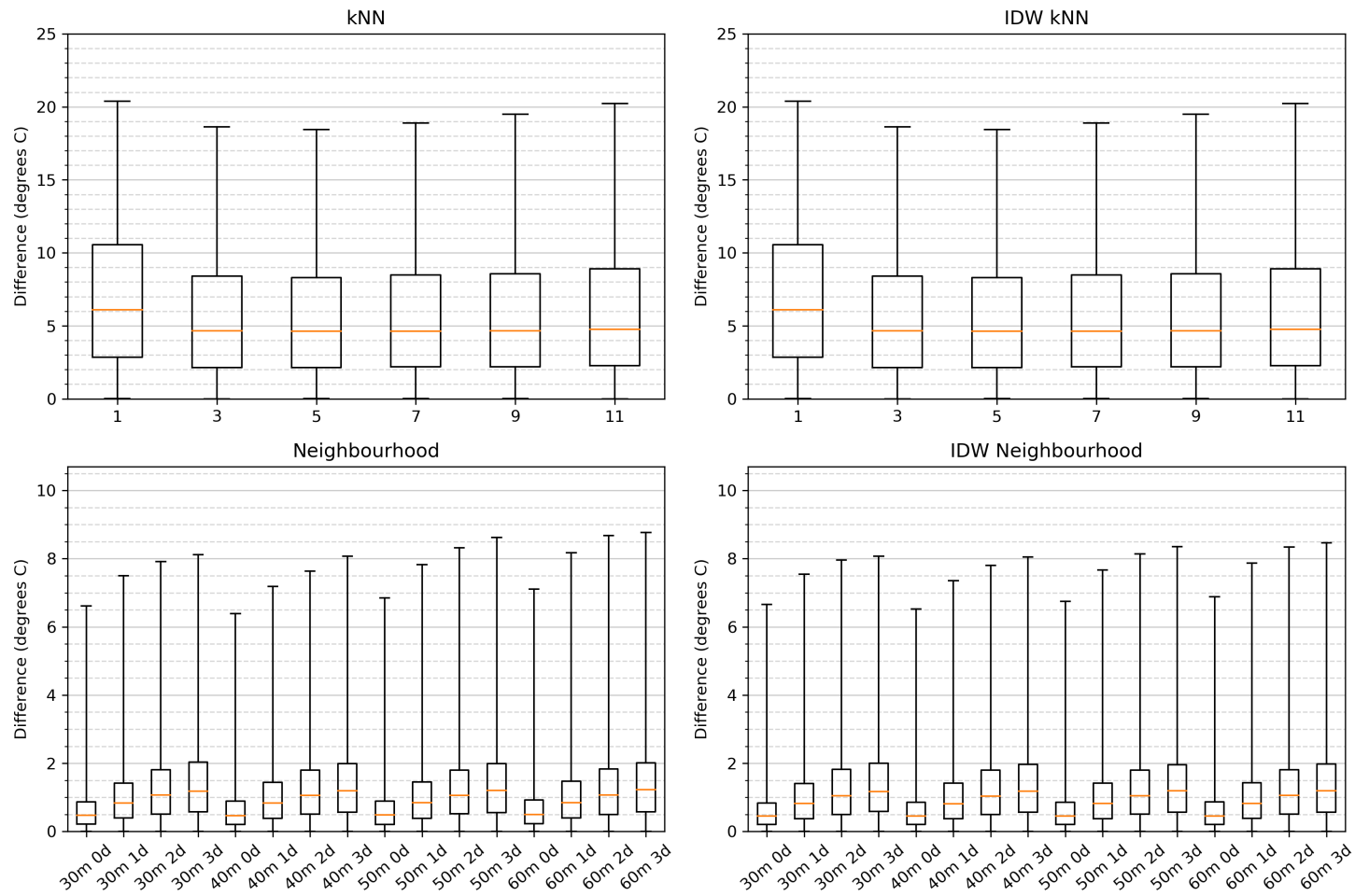


Figure B.2: Accuracy of imputing withheld instances for the temperature dataset

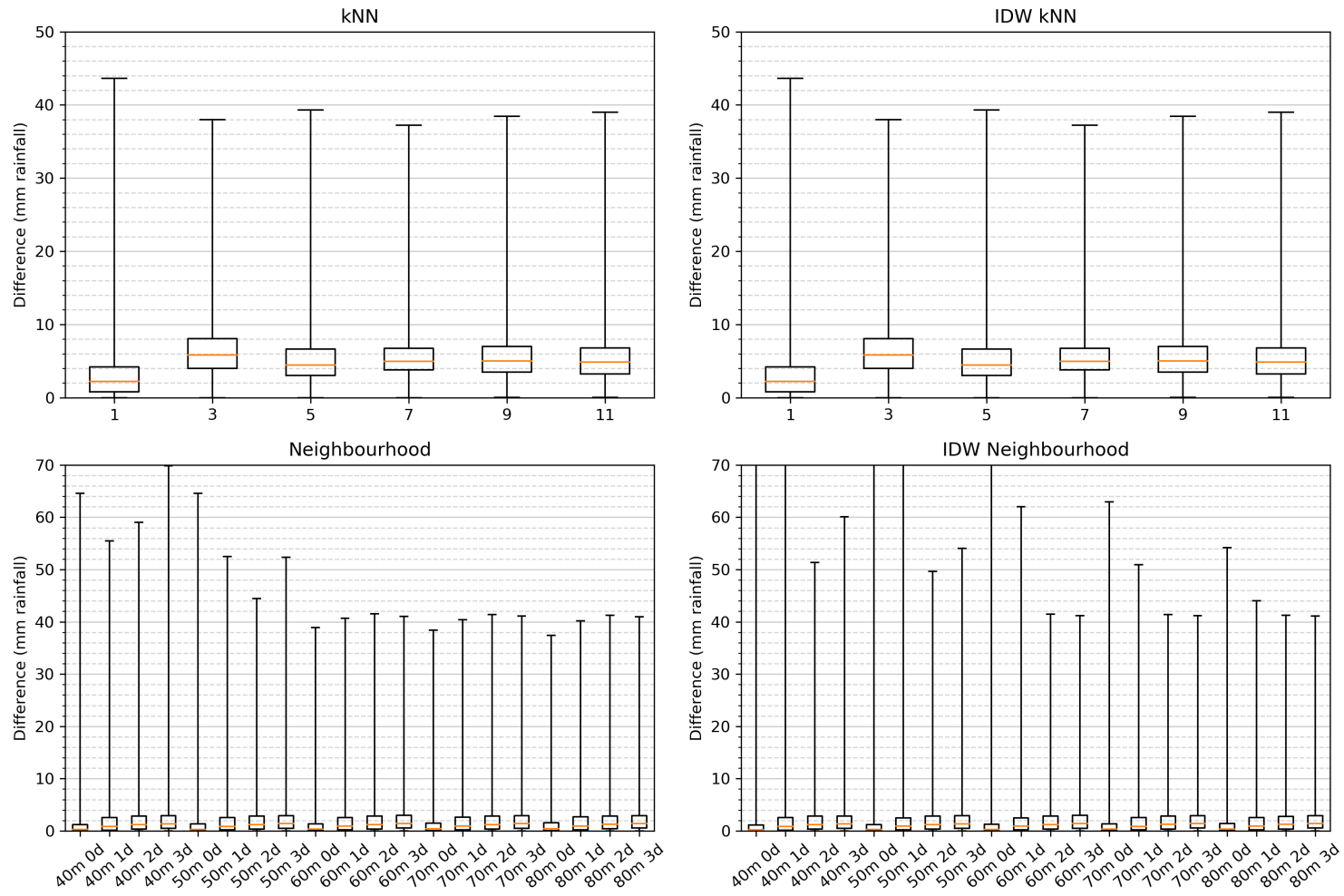


Figure B.3: Accuracy of imputing withheld instances for the rainfall dataset

## Appendix C

# Supplementary Results for Chapter 5

The results in this appendix complement the discussion of results presented in Section 5.5.

### C.1 Speedup Achieved Using Reduced Datasets

These figures show the time taken to augment the primary dataset using the raw supplementary datasets using the kNN, IDW kNN, neighbourhood and IDW neighbourhood methods (denoted *Baseline*). Furthermore, they show the time taken to augment the primary dataset using the reduced supplementary datasets, after they have been reduced using the RES (Chapter 4) and LES (Chapter 5) heuristics.

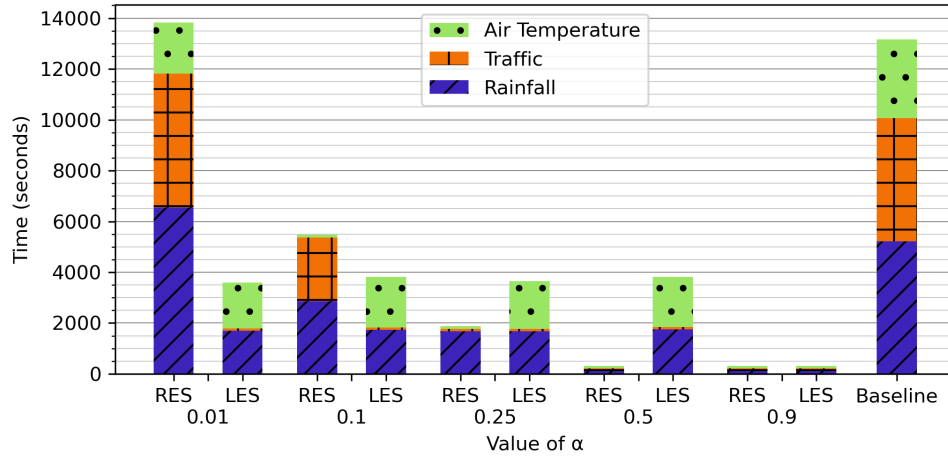


Figure C.1: Time required for linking compared to kNN baseline.

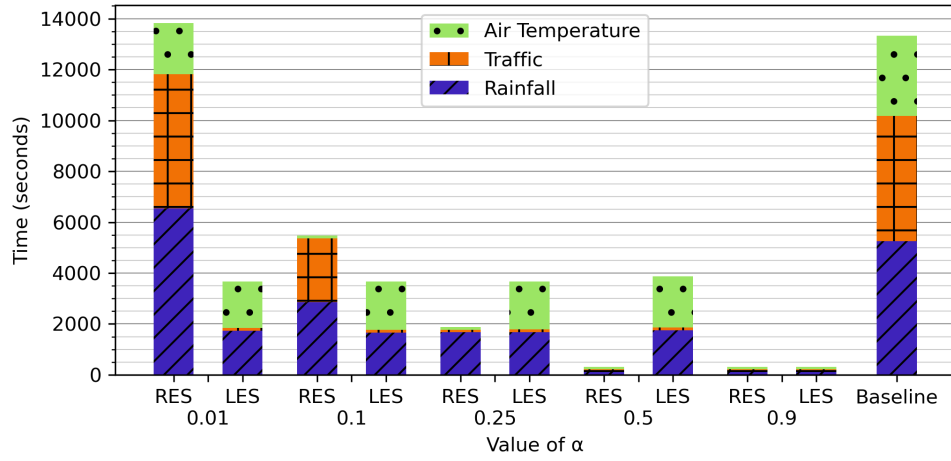


Figure C.2: Time required for linking compared to IDW kNN baseline.

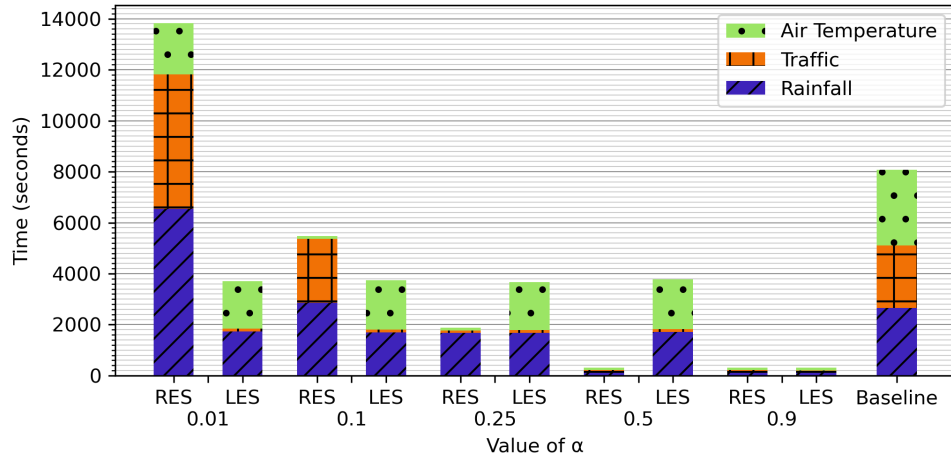


Figure C.3: Time required for linking compared to Neighbourhood baseline.

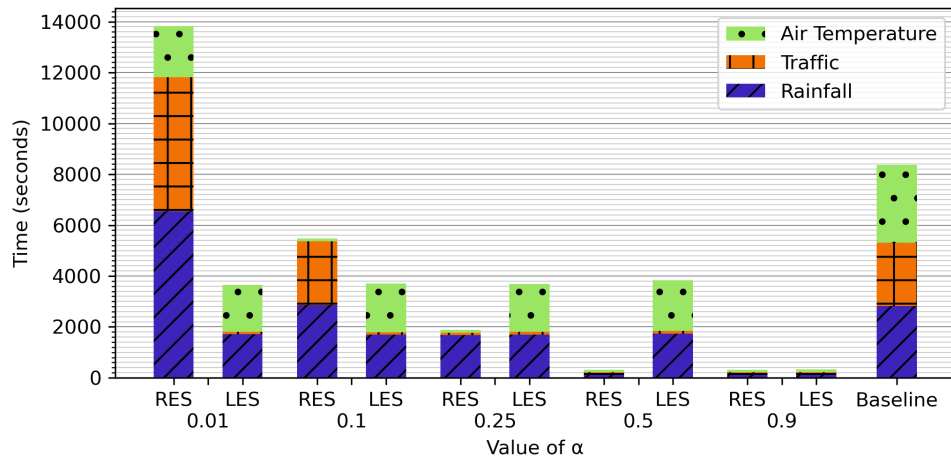


Figure C.4: Time required for linking compared to IDW Neighbourhood baseline.

## C.2 Error in Linked Features

These results show the absolute percentage error in the linked features created using reduced supplementary datasets (reduced using the LES heuristic). Results are shown for each of the four linking methods used in the LES heuristic.

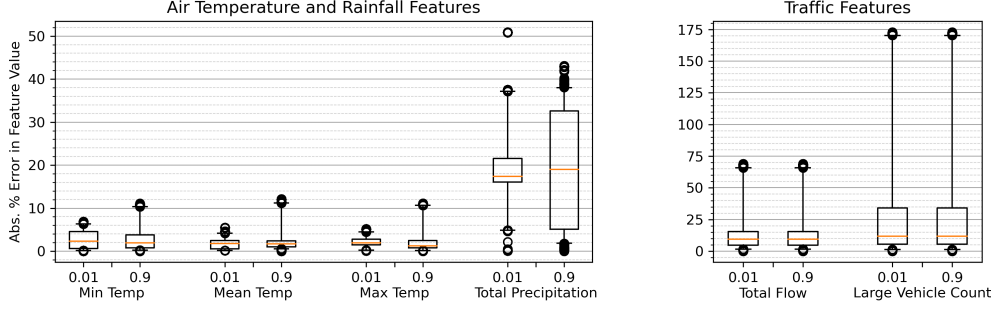


Figure C.5: Error incurred by engineering features using the reduced datasets (created using the LES heuristic with the kNN linking method).

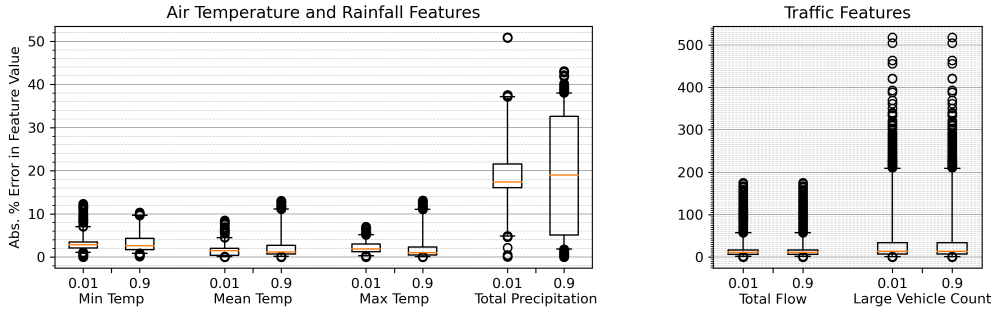


Figure C.6: Error incurred by engineering features using the reduced datasets (created using the LES heuristic with the IDW kNN linking method).

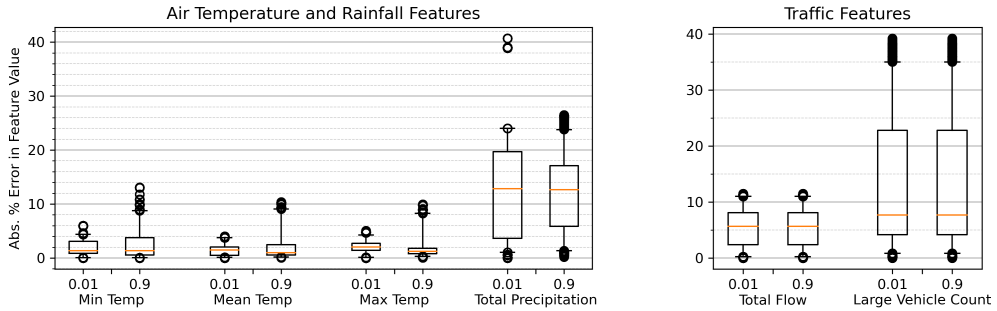


Figure C.7: Error incurred by engineering features using the reduced datasets (created using the LES heuristic with the neighbourhood linking method).

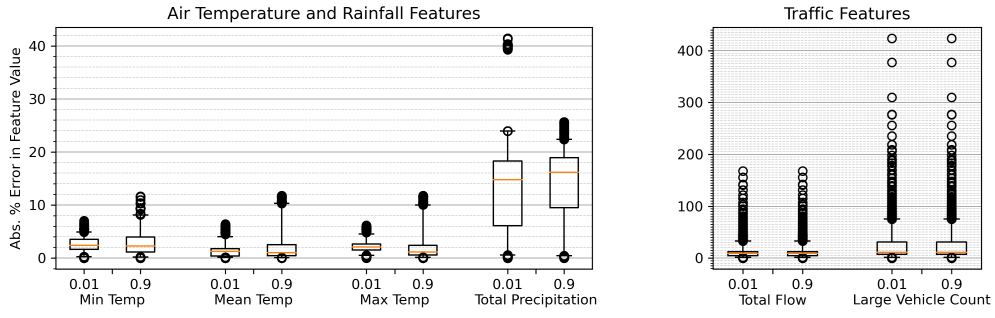


Figure C.8: Error incurred by engineering features using the reduced datasets (created using the LES heuristic with the IDW neighbourhood linking method).

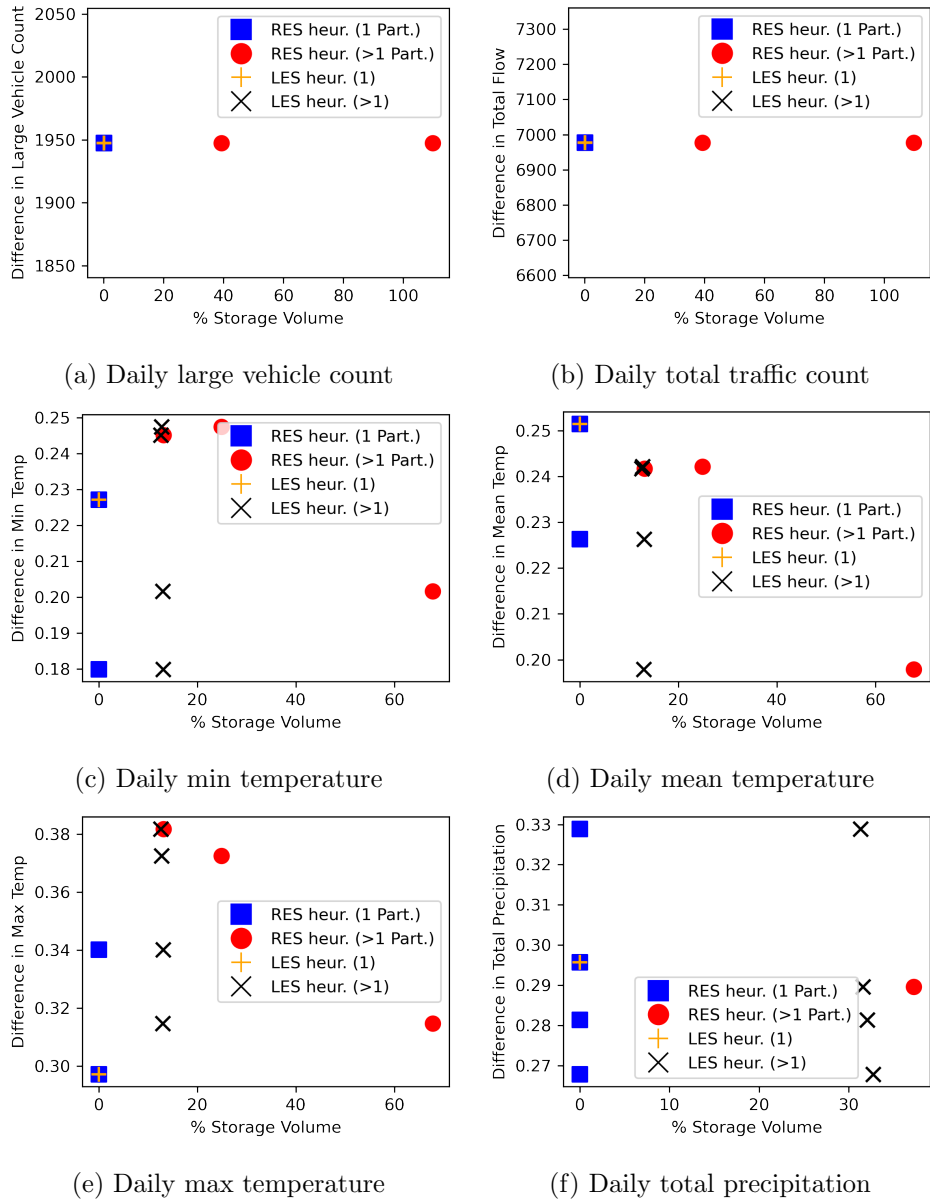


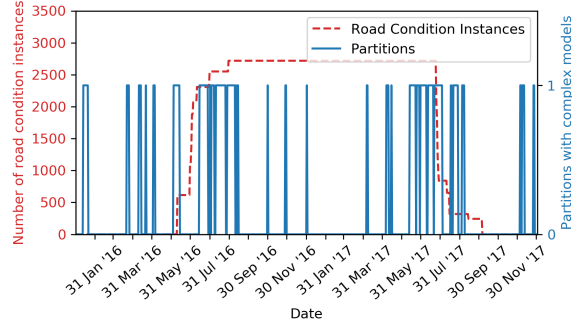
Figure C.9: Error in engineered features versus storage used by the reduced datasets created using the RES and LES heuristics.

### C.3 Retention of Information in Space and Time

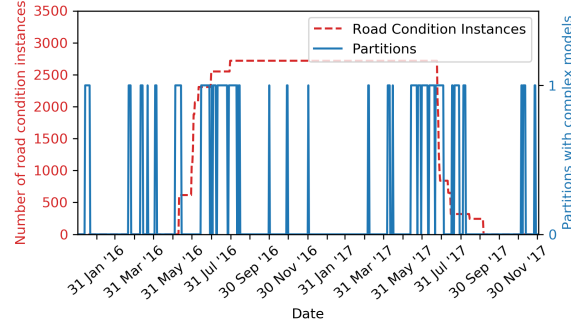
These results show the number of partitions in the reduced datasets after the supplementary datasets have been reduced using the RES and LES heuristics. As shown, the RES heuristic frequently stores models with more than 1 model coefficient that are not applicable to the primary dataset (i.e. they do not fall within the time period May 2016–September 2017). In contrast, the LES heuristic only stores more than 1 model coefficient for partitions that overlap with the primary dataset’s instances (i.e. the time period May 2016–September 2017).

Results are shown for when  $\alpha = 0.1$ , though results for all values of  $\alpha$  may be found in the online repository [120]. When  $\alpha > 0.5$ , the number of partitions output for each dataset was always 1, and this partition covered the entire spatial and temporal domains.

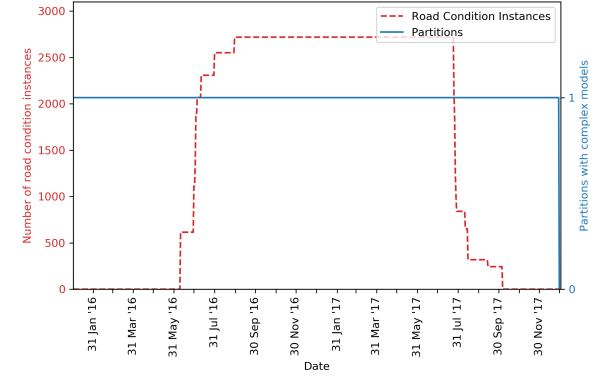




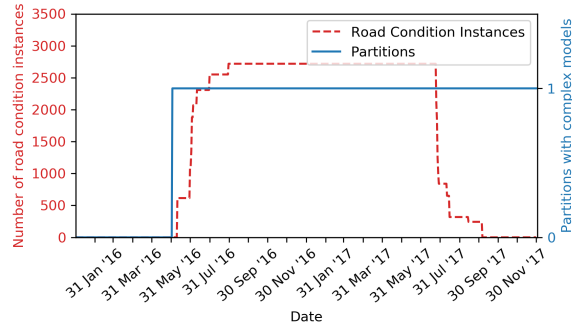
(a) Traffic dataset, RES



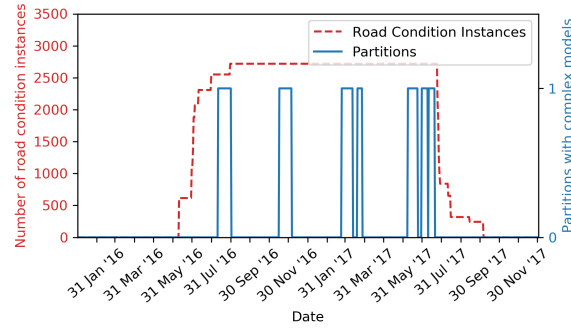
(b) Air temperature dataset, RES



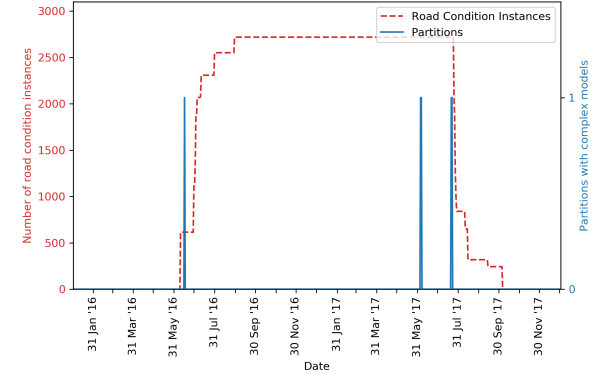
(c) Rainfall dataset, RES



(d) Traffic dataset, LES



(e) Air temperature dataset, LES



(f) Rainfall dataset, LES

Figure C.10: Temporal locations of partitions with more than one model coefficient when  $\alpha = 0.1$ .

## C.4 Error in Original Supplementary Features

The following results show the error incurred in the original features versus storage used by the reduced datasets when reduced using the RES and LES heuristics.

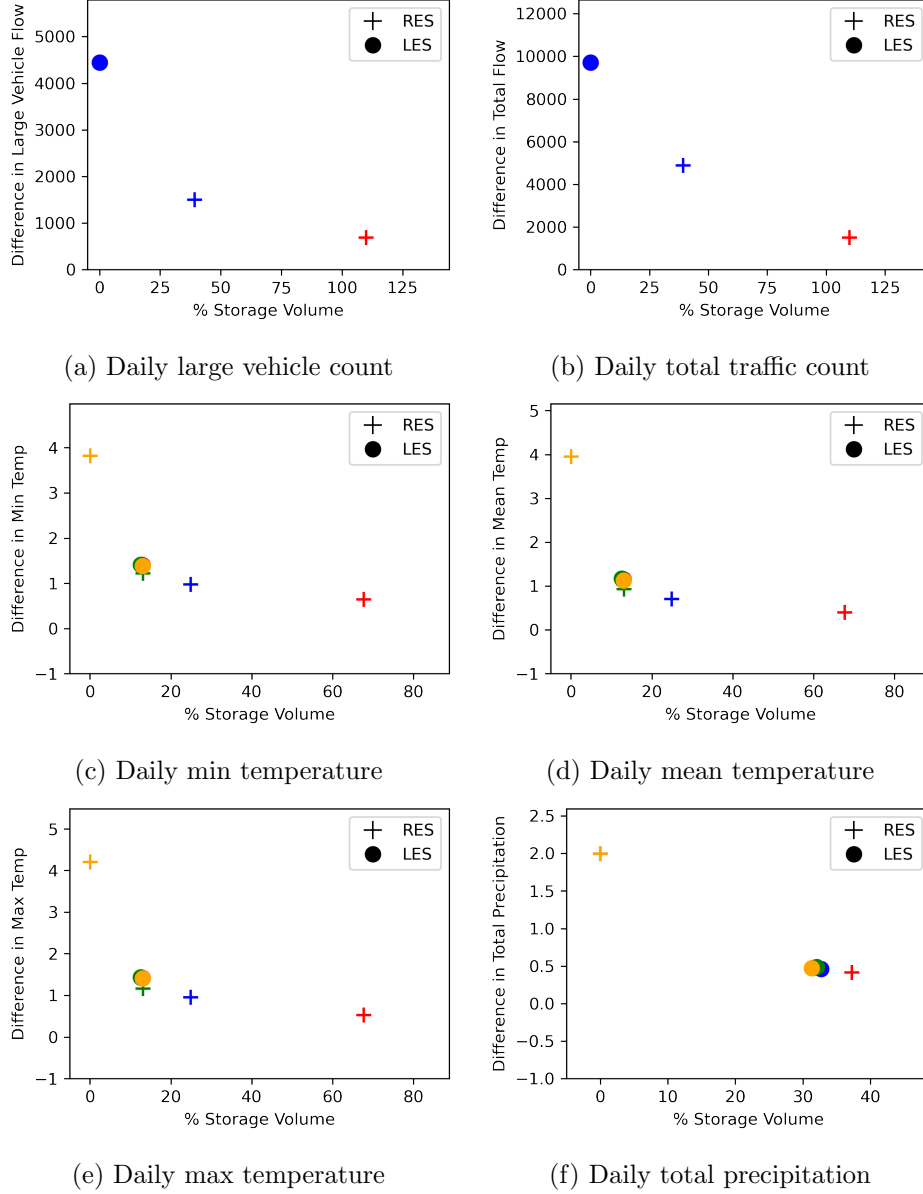


Figure C.11: Error in original supplementary features versus storage used by the reduced datasets created using the RES and LES heuristics.

## Appendix D

# Supplementary Results for Section 6.1

This appendix complements the discussion presented in Section 6.1.3.

### D.1 Effect on Storage Used

These results show the storage used when independently and mutually reducing datasets.

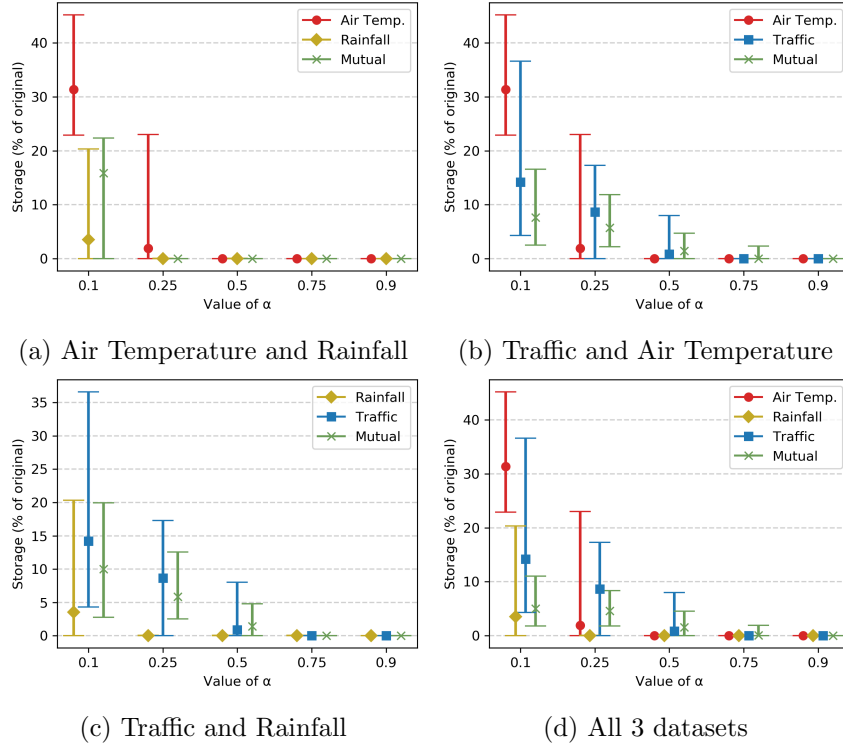


Figure D.1: Total storage used by each of the independently reduced datasets (reduced using  $k$ D-STR) versus the storage used by the mutually reduced dataset (reduced using  $Mk$ D-STR, denoted *Mutual*).

## D.2 Effect on Error Incurred

These results show the error incurred when mutually reducing the datasets using *MkD-STR* versus reducing them independently using *kD-STR*.

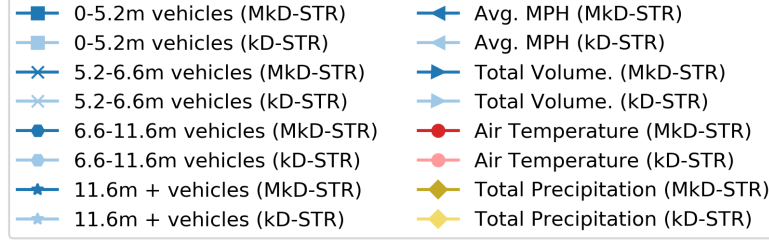


Figure D.2: Legend used for the following two figures. Results are shown for each feature in the 3 datasets tested — results for the traffic dataset are shown in blue, results for the air temperature dataset are shown in red and results for the rainfall dataset are shown in yellow. Darker colours are used to denote results for *MkD-STR* and lighter colours are used to denote results for *kD-STR*.

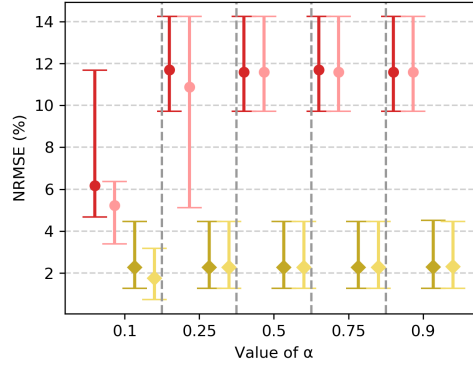
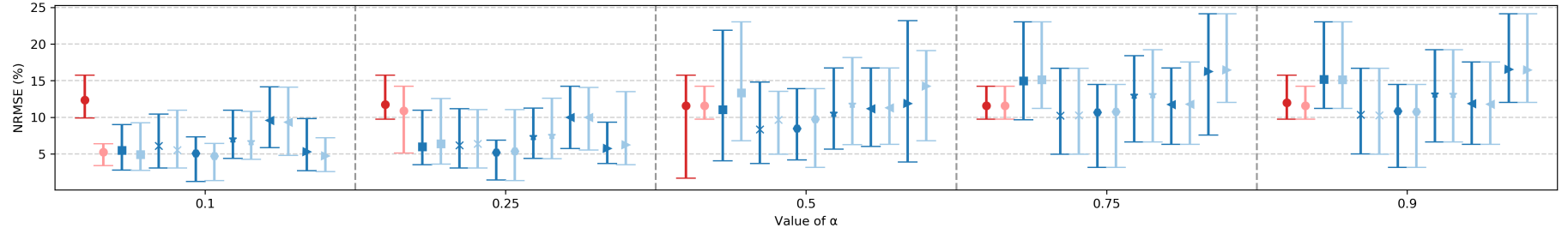
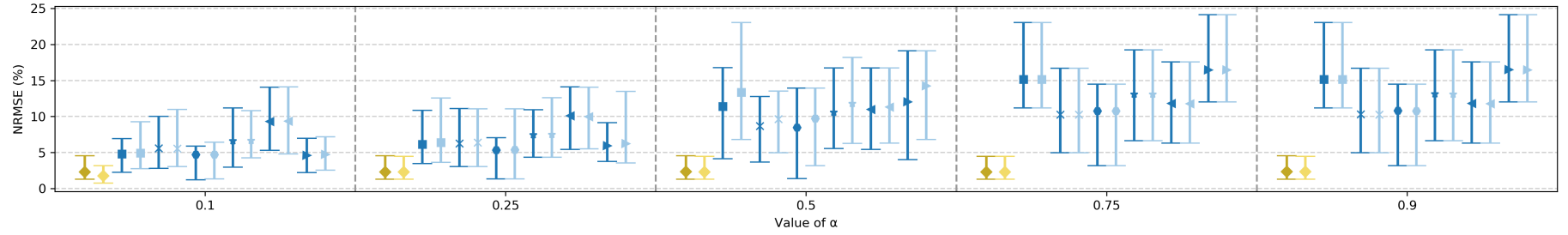


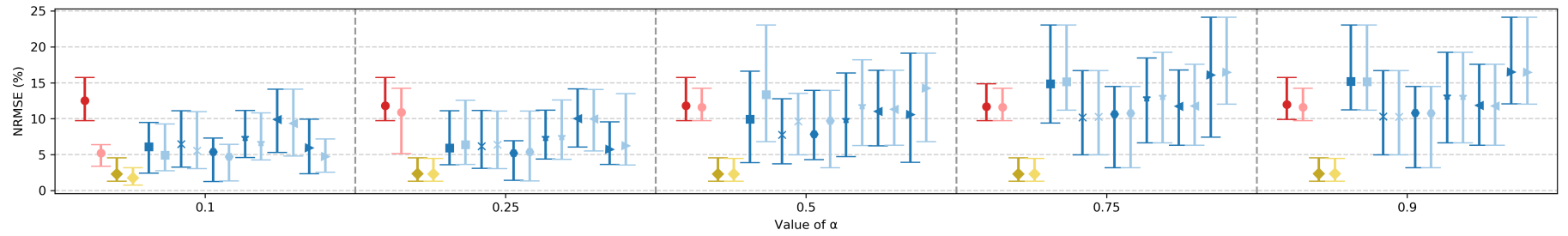
Figure D.3: NRMSE incurred by *kD-STR* versus *MkD-STR* when reducing the air temperature and rainfall datasets. The legend for this figure is shown above, as well as Figure 6.5.



(a) Traffic and Air Temperature



(b) Traffic and Rainfall



(c) All 3 datasets

Figure D.4: NRMSE incurred by  $k$ D-STR versus  $Mk$ D-STR. The legend for this figure is shown on the previous page, as well as Figure 6.5.

## Appendix E

# Survey of Distributed Hierarchical Clustering Techniques

In Chapter 4, a spatio-temporal partitioning method was introduced that clusters instances in the feature space then, for each level of the resulting hierarchy, discovers connected components in the spatio-temporal (ST) space of each cluster. Each connected component then constitutes a partition in the ST space. However, this method is limited in scale by the need for computing the complete distance matrix between all instances for hierarchical clustering. To overcome this issue, distributed hierarchical clustering methods have been proposed.

Naive approaches have been proposed that approximate hierarchical clustering in distributed environments. For example, Woodley *et al.* adapted the KD-tree partitioning approach to output the representatives used at each level of the tree as a divisive hierarchical clustering [138]. After creating a KD-tree, the method cuts the tree at level  $l$  in order to find  $k^{l-1}$  cluster centers, and each instance is then assigned to its nearest cluster center. This method provides an approximate hierarchical clustering, however the number of partitions returned can only be a multiple of  $k$ . Since the cluster tree is built during the ingestion of instances, splitting a partition into a further  $k$  partitions can be performed both locally on the compute node containing that cluster, and simply requires propagating the cluster ID down the tree to the individual instances at the leaf nodes. When  $k = 2$ , this process is similar to an agglomerative hierarchical clustering using Ward's criterion, the linking criterion used in Chapter 4.

Less naive approaches for hierarchical clustering focus on single linkage agglomerative clustering. In [62], the authors show the equivalence between calculating single linkage clustering and finding the minimum spanning tree (MST) of the data in the feature space. Jin *et al.* use this to perform single

linkage clustering [68, 69]. First, their algorithm splits the dataset into  $x$  equal-sized partitions in the feature space, then creates a subgraph for each partition and each of the  $\binom{2}{x}$  possible partition pairings, a technique referred to as *random blocks partitioning*. Each subgraph is then processed in parallel, using Prim’s algorithm to find the MST on each subgraph. Finally, the MST of each  $K$  subgraphs is combined recursively using Kruskal’s algorithm, where  $K$  is a user-defined number. When the algorithm terminates, the final MST is calculated. To partition the dataset into a set of  $n$  clusters, the list of edges in the MST is sorted in descending order and the first  $n - 1$  edges cut to form  $n$  distinct components. Similar approaches have been taken in other algorithms [16].

However, the random blocks partitioning approach has been criticised as an expensive solution for calculating hierarchical clustering [107]. Instead, Santos *et al.* use a recursive technique that initially places all instances in a single partition, and samples the data, using each sample as the center of a new partition [107]. Each instance is then assigned to the partition with the closest center, and each partition is then recursively partitioned until each partition can fit in the memory of a single compute node. Then, the MST is calculated for each partition and finally the edges from each partition’s MST combined into a single MST. Whilst the initial partitioning of the data is similar to the construction of a kd-tree used multiple distributed clustering algorithms, this approach is also similar to the spectral clustering approach taken in [143]. Again, this approach is likely to suffer the same sensitivity to the number of representatives chosen at each stage of partitioning.

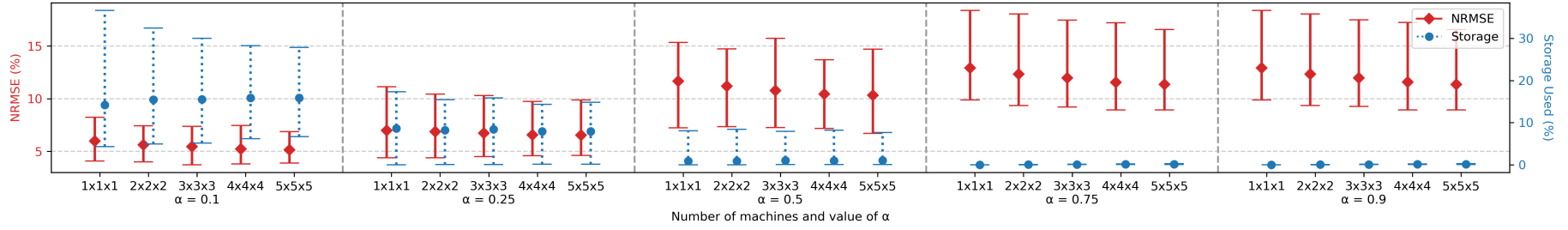
In a more general technique, the DHC algorithm uses a KD-tree to store the original data for fast nearest-neighbour retrieval [146]. Each partition on the KD-tree first calculates the maximum distance of instances from the partition boundary it requires from each of its neighbouring partitions for nearest-neighbour calculations. Then, each partition requests instances from its neighbours within that distance of the boundary, and uses these returned instances to calculate the global nearest neighbour of each of its own instances. By then repeating a process of merging nearest neighbours, clusters form in the feature space. Furthermore, different linkage functions can be used to merge candidate clusters. This agglomerative approach would be useful in the feature space, though  $\mathcal{O}(n)$  storage is required to store the nearest neighbour information of each instance and the number of iterations may still be  $\mathcal{O}(n)$  despite being around  $\mathcal{O}(\log n)$  in many scenarios. Further, significant communication is needed to calculate the distance between clusters as clusters grow in size and span multiple compute nodes.

## Appendix F

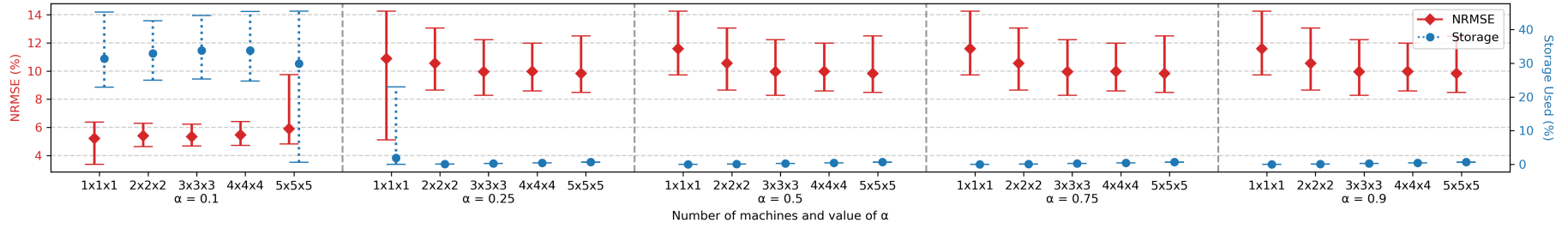
# Supplementary Results for Section 6.2

The results in this appendix complement the discussion of results presented in Section 6.2.3. These results show the storage used and reconstruction error incurred by D $k$ D-STR for the traffic, air temperature and rainfall datasets.

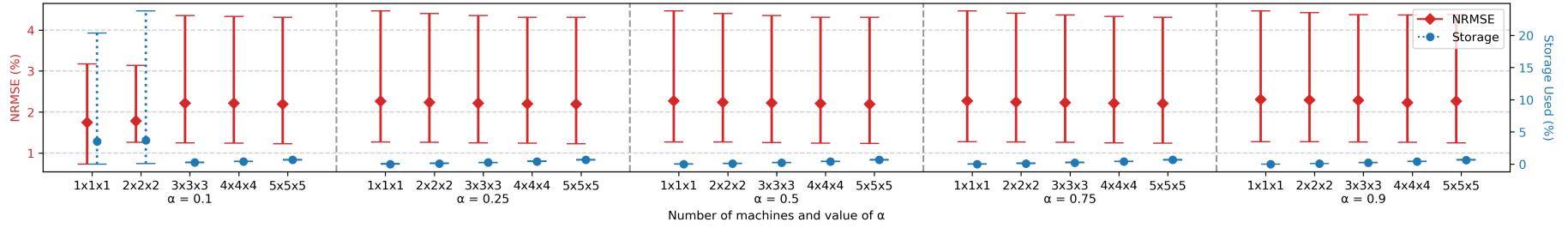




(a) Traffic



(b) Air Temperature



(c) Rainfall

Figure F.1: Storage used and NRMSE incurred by  $DkD$ -STR for the datasets presented in Chapter 3.

## Appendix G

# Selecting Data Samples for Perturbations

It is desirable for the data samples used for perturbations analysis (Section 6.3) to be as representative as possible of the underlying spatio-temporal processes, though few samples could be tested. Therefore, 1 sample was used for each of the air temperature, rainfall, A-road and M-road datasets<sup>1</sup>. It was the seasonality and periodicity can reasonably be assumed to be approximately the same for each month for each of the datasets.

The data samples used needed to be the most stationary to ensure changes in the reduced output are attributable to the perturbations of the input data. To test for spatial stationariness, Moran's I statistic was used. Each of the data samples tested presented I values near to 0, indicating little spatial correlation between instances recorded at the same time. For example, the mean I value for the *Total Volume* feature of the M1 samples was 0.107 which was far from the +1 and -1 values expected for strong spatial correlation. In the temporal domain, the Dickey-Fuller test was used, wherein strong temporal stationariness was found for all of the data samples. For example, the mean Dickey-Fuller test value for the *Total Volume* feature of the M1 samples was -9.156, far below the mean 5% value of -2.862 used to confirm stationariness.

As well as choosing the most stationary data samples, the samples needed to be representative of the spatio-temporal distribution of instances within a dataset, as well as representative of the distribution of feature values. For the air temperature dataset, all 12 months were found to have similar distributions of sensors in space as shown in Figure G.1. Similarly, the distribution of blackouts were also shown to be similar, as seen in Figure G.2. In both figures, October is shown to have a slightly different distribution as it experienced a greater number of shorter blackouts and its sensors were closer together

---

<sup>1</sup>As the characteristics of A-roads and M-roads differs within England, 1 sample from each road type was tested.

than the other months. However, the shape of the distributions for all months were reasonably similar. In the feature space, some months had quite narrow Gaussian distributions as shown in Figure G.3. Other months exhibited wider distributions that were bimodal. These months also exhibited a higher variance in space than the summer months.

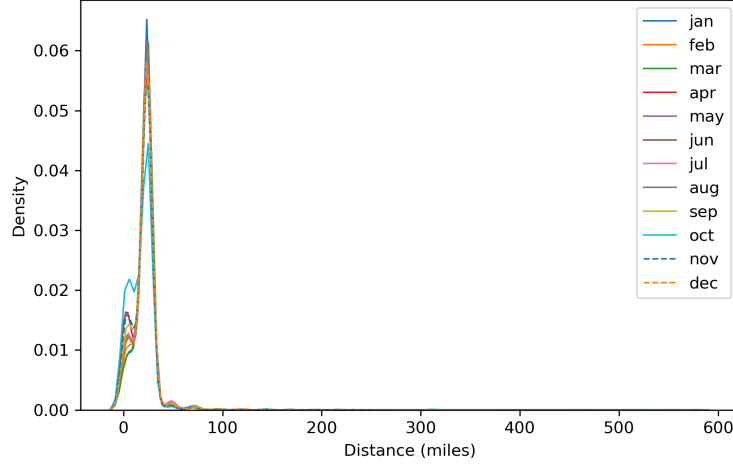


Figure G.1: Distances between each sensor and its nearest neighbour in the 12 1-month samples of air temperature data.

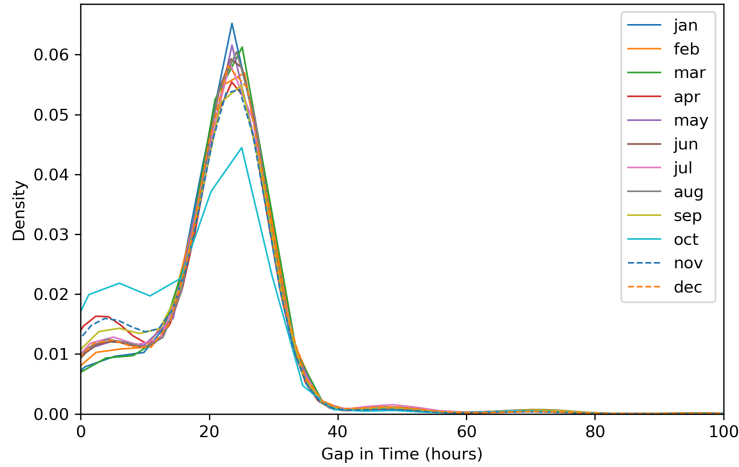


Figure G.2: Lengths of time for which each sensor failed to record multiple consecutive instances in each of the 12 1-month samples of air temperature data.

For the rainfall dataset, the distribution of sensors in space was similar across all samples. In the temporal domain, some months exhibited longer periods of inactivity than others, however the mean and median lengths of blackout for all of the samples was approximately the same for all months. Like the air temperature dataset, some months exhibited wider distributions of feature values than others. This was correlated with higher variance in both the spatial and temporal domains.

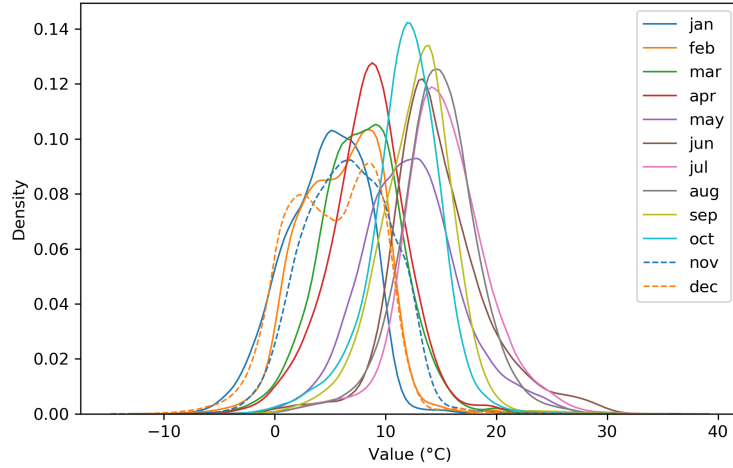
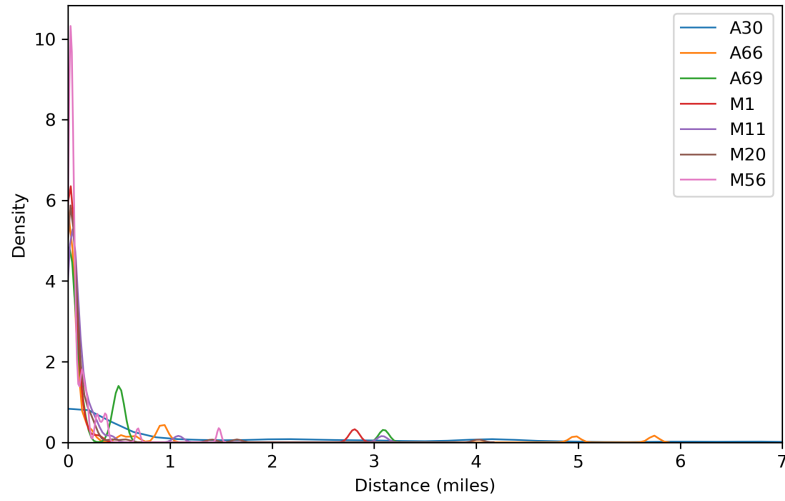


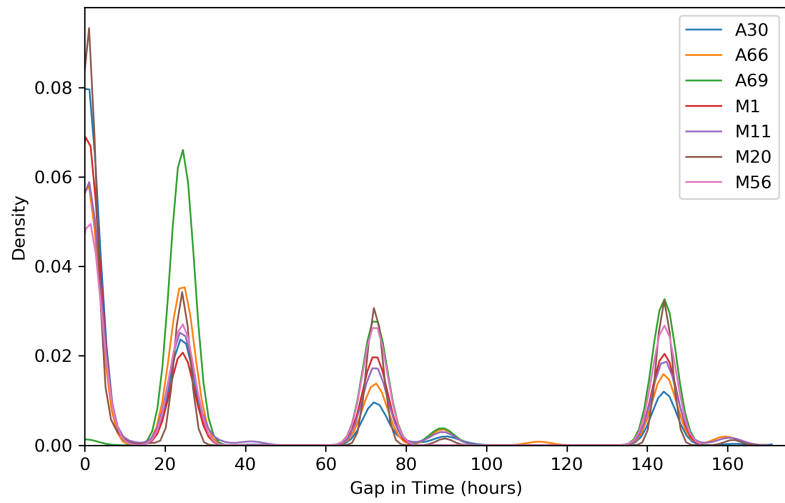
Figure G.3: Distribution of feature values for each of the 12 1-month samples of air temperature data.

For the traffic datasets, the distribution of distances between sensors was approximately the for all samples on the same road. However, each of the roads exhibited different spatial distributions compared to each of the other roads. For example, in Figure G.4(a), all roads have a peak distance between sensors of close to 0 miles. However, each road exhibits different small peaks beyond the initial peak close to 0 miles. Furthermore, in Figure G.4(b), each road is shown to exhibit different lengths of time for which a sensor did not record an instance. Though the peaks in Figure G.4(b) occur at the same points in time, the height each road within each peak is different. Furthermore, the distribution of traffic counts for 4 different vehicle sizes is different between each of the roads, as is the distribution of average speeds. Thus, the A30 and M1 roads were chosen for their different characteristics, and that each of the other roads demonstrated similar characteristics to these two roads.

Thus, for the traffic dataset, the A30 and M1 roads were selected as representative roads. For each of the four selected datasets (A30 traffic, M1 traffic, air temperature and rainfall), each of the month-long data samples were equally valid for the perturbations analysis undertaken in Section 6.3 according to their stationariness. However, the samples that presented both the lowest Moran's I and Dickey-Fuller results were selected. These choices were also validated with domain experts at TRL who confirmed these data samples were typical of the data collected for each of the data sources, and were suitable choices for the perturbations analysis presented in Section 6.3.



(a) Distance between nearest sensors in space



(b) Consecutive periods of missing data

Figure G.4: Distances between sensors for all roads in May 2017, the lengths of time for which a sensor did not record an instance for all roads in May 2017.

## Appendix H

# Supplementary Results for Section 6.3

The results in this appendix complement the discussion of results presented in Section 6.3.3.

### H.1 Effect on Variance

In this appendix, the variance for the four datasets used in Section 6.3 are presented. For each, the variance of the raw clean dataset, the raw perturbed dataset (used as input for  $k$ D-STR), and the reconstructed perturbed data from the reduced form are presented.

### H.2 Effect on Partitioning

These results show the number of partitions in the reduced datasets after they have been perturbed in space and time, noise has been added to the feature values, and increasing amounts of instances are missing

Table H.1: Noisy data: variance of the raw clean datasets, raw perturbed datasets and reduced perturbed datasets. Results are shown for DTR modelling on partitions. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

$\alpha$	SD.	Clean	A30 Pert.	Recon.	Clean	M1 Pert.	Recon.	Clean	Air Temperature Pert.	Recon.	Clean	Rainfall Pert.	Recon.
0.1	0.1	21130	21336	19535	235927	238288	227207	11.1	11.21	9.01	0.272	0.275	0.091
	0.5	21130	26412	21160	235927	295534	239893	11.1	13.85	9.36	0.272	0.34	0.081
	1.0	21130	42082	23179	235927	470907	286244	11.1	22.23	10.34	0.272	0.545	0.089
	1.5	21130	68981	38020	235927	768325	463290	11.1	36.26	12.06	0.272	0.883	0.087
	2.0	21130	105365	55406	235927	1177074	738115	11.1	55.43	12.93	0.272	1.364	0.101
0.25	0.1	21130	21336	6219	235927	238288	218566	11.1	11.21	8.52	0.272	0.275	0.055
	0.5	21130	26412	6281	235927	295534	84145	11.1	13.85	8.33	0.272	0.34	0.049
	1.0	21130	42082	6099	235927	470907	83486	11.1	22.23	8.55	0.272	0.545	0.038
	1.5	21130	68981	5326	235927	768325	81791	11.1	36.26	8.43	0.272	0.883	0.042
	2.0	21130	105365	5075	235927	1177074	84344	11.1	55.43	8.65	0.272	1.364	0.044
0.5	0.1	21130	21336	5088	235927	238288	79146	11.1	11.21	8.23	0.272	0.275	0.03
	0.5	21130	26412	4833	235927	295534	77454	11.1	13.85	7.94	0.272	0.34	0.026
	1.0	21130	42082	4440	235927	470907	75057	11.1	22.23	7.55	0.272	0.545	0.019
	1.5	21130	68981	4151	235927	768325	74306	11.1	36.26	7.17	0.272	0.883	0.018
	2.0	21130	105365	4159	235927	1177074	73732	11.1	55.43	7.16	0.272	1.364	0.017
0.75	0.1	21130	21336	3853	235927	238288	75418	11.1	11.21	7.46	0.272	0.275	0.01
	0.5	21130	26412	3745	235927	295534	75696	11.1	13.85	6.9	0.272	0.34	0.009
	1.0	21130	42082	3901	235927	470907	72705	11.1	22.23	7.04	0.272	0.545	0.005
	1.5	21130	68981	3133	235927	768325	72012	11.1	36.26	6.59	0.272	0.883	0.005
	2.0	21130	105365	3035	235927	1177074	72639	11.1	55.43	5.85	0.272	1.364	0.001
0.9	0.1	21130	21336	3853	235927	238288	71286	11.1	11.21	6.95	0.272	0.275	0.005
	0.5	21130	26412	3471	235927	295534	71848	11.1	13.85	6.22	0.272	0.34	0.001
	1.0	21130	42082	3429	235927	470907	67806	11.1	22.23	5.77	0.272	0.545	0.001
	1.5	21130	68981	2440	235927	768325	68659	11.1	36.26	5.31	0.272	0.883	0.001
	2.0	21130	105365	2204	235927	1177074	62703	11.1	55.43	5.22	0.272	1.364	0.001

Table H.2: Missing at random: variance of the raw clean datasets, raw perturbed datasets and reduced perturbed datasets. Results are shown for DTR modelling on partitions. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

$\alpha$	SD.	Clean	A30 Pert.	Recon.	Clean	M1 Pert.	Recon.	Clean	Air Temperature Pert.	Recon.	Clean	Rainfall Pert.	Recon.
0.1	10%	21130	21123	19804	235927	235974	222747	11.1	11.11	8.97	0.272	0.27	0.181
	20%	21130	21141	19513	235927	235909	221788	11.1	11.09	9.0	0.272	0.273	0.187
	30%	21130	21145	19408	235927	235866	220897	11.1	11.09	9.02	0.272	0.275	0.193
	40%	21130	21062	19697	235927	235879	226683	11.1	11.11	9.05	0.272	0.266	0.266
	50%	21130	21101	19663	235927	236703	222932	11.1	11.1	9.06	0.272	0.277	0.198
0.25	10%	21130	21123	6518	235927	235974	218549	11.1	11.11	8.51	0.272	0.27	0.067
	20%	21130	21141	6138	235927	235909	222170	11.1	11.09	8.53	0.272	0.273	0.055
	30%	21130	21145	6374	235927	235866	152916	11.1	11.09	8.54	0.272	0.275	0.059
	40%	21130	21062	6175	235927	235879	88157	11.1	11.11	8.43	0.272	0.266	0.057
	50%	21130	21101	5947	235927	236703	153791	11.1	11.1	8.29	0.272	0.277	0.061
0.5	10%	21130	21123	5385	235927	235974	75505	11.1	11.11	7.89	0.272	0.27	0.034
	20%	21130	21141	5039	235927	235909	77710	11.1	11.09	7.9	0.272	0.273	0.031
	30%	21130	21145	5394	235927	235866	77564	11.1	11.09	7.88	0.272	0.275	0.028
	40%	21130	21062	4637	235927	235879	78433	11.1	11.11	7.93	0.272	0.266	0.036
	50%	21130	21101	4523	235927	236703	79467	11.1	11.1	7.94	0.272	0.277	0.028
0.75	10%	21130	21123	4479	235927	235974	75839	11.1	11.11	7.43	0.272	0.27	0.009
	20%	21130	21141	4136	235927	235909	76227	11.1	11.09	7.44	0.272	0.273	0.011
	30%	21130	21145	4177	235927	235866	74282	11.1	11.09	7.47	0.272	0.275	0.012
	40%	21130	21062	4152	235927	235879	75127	11.1	11.11	7.48	0.272	0.266	0.009
	50%	21130	21101	3834	235927	236703	76788	11.1	11.1	6.94	0.272	0.277	0.005
0.9	10%	21130	21123	4021	235927	235974	71550	11.1	11.11	6.93	0.272	0.27	0.001
	20%	21130	21141	3826	235927	235909	72262	11.1	11.09	6.94	0.272	0.273	0.005
	30%	21130	21145	3854	235927	235866	72158	11.1	11.09	6.28	0.272	0.275	0.005
	40%	21130	21062	3477	235927	235879	72913	11.1	11.11	6.29	0.272	0.266	0.001
	50%	21130	21101	3414	235927	236703	73915	11.1	11.1	6.04	0.272	0.277	0.001

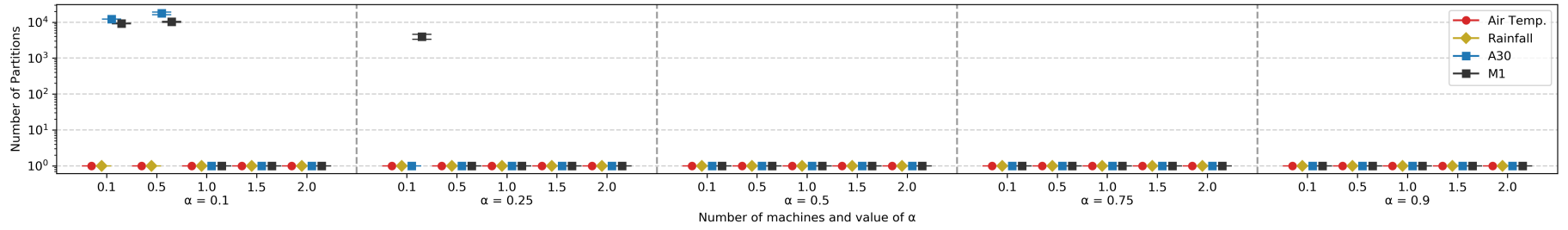
Table H.3: Missing not at random: variance of the raw clean datasets, raw perturbed datasets and reduced perturbed datasets. Results are shown for DTR modelling on partitions. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

$\alpha$	SD.	Clean	A30 Pert.	Recon.	Clean	M1 Pert.	Recon.	Air Temperature			Clean	Rainfall Pert.	Recon.
0.1	10%	21130	21124	19637	235927	235857	220754	11.1	11.1	8.98	0.272	0.271	0.19
	20%	21130	21152	19314	235927	235896	224250	11.1	11.12	9.03	0.272	0.273	0.196
	30%	21130	21094	19439	235927	235996	220493	11.1	11.12	9.01	0.272	0.266	0.188
	40%	21130	21211	19638	235927	236266	218911	11.1	11.12	9.05	0.272	0.275	0.271
	50%	21130	21135	19954	235927	236232	224798	11.1	11.1	9.04	0.272	0.276	0.203
0.25	10%	21130	21124	6449	235927	235857	220119	11.1	11.1	8.5	0.272	0.271	0.069
	20%	21130	21152	6309	235927	235896	219800	11.1	11.12	8.55	0.272	0.273	0.071
	30%	21130	21094	6143	235927	235996	152223	11.1	11.12	8.53	0.272	0.266	0.069
	40%	21130	21211	6254	235927	236266	218613	11.1	11.12	8.42	0.272	0.275	0.067
	50%	21130	21135	6059	235927	236232	83681	11.1	11.1	8.28	0.272	0.276	0.061
0.5	10%	21130	21124	5340	235927	235857	79039	11.1	11.1	7.86	0.272	0.271	0.031
	20%	21130	21152	5350	235927	235896	76809	11.1	11.12	7.91	0.272	0.273	0.031
	30%	21130	21094	4744	235927	235996	76991	11.1	11.12	7.9	0.272	0.266	0.03
	40%	21130	21211	4505	235927	236266	78672	11.1	11.12	7.92	0.272	0.275	0.032
	50%	21130	21135	4505	235927	236232	79023	11.1	11.1	7.91	0.272	0.276	0.026
0.75	10%	21130	21124	3865	235927	235857	75199	11.1	11.1	7.42	0.272	0.271	0.01
	20%	21130	21152	3697	235927	235896	75504	11.1	11.12	7.44	0.272	0.273	0.01
	30%	21130	21094	4156	235927	235996	73620	11.1	11.12	7.42	0.272	0.266	0.006
	40%	21130	21211	3961	235927	236266	73917	11.1	11.12	7.2	0.272	0.275	0.012
	50%	21130	21135	3737	235927	236232	74285	11.1	11.1	6.94	0.272	0.276	0.008
0.9	10%	21130	21124	3865	235927	235857	71312	11.1	11.1	6.9	0.272	0.271	0.005
	20%	21130	21152	3428	235927	235896	71321	11.1	11.12	6.95	0.272	0.273	0.005
	30%	21130	21094	3828	235927	235996	71462	11.1	11.12	6.26	0.272	0.266	0.005
	40%	21130	21211	3482	235927	236266	71827	11.1	11.12	6.28	0.272	0.275	0.001
	50%	21130	21135	3478	235927	236232	72047	11.1	11.1	5.77	0.272	0.276	0.001

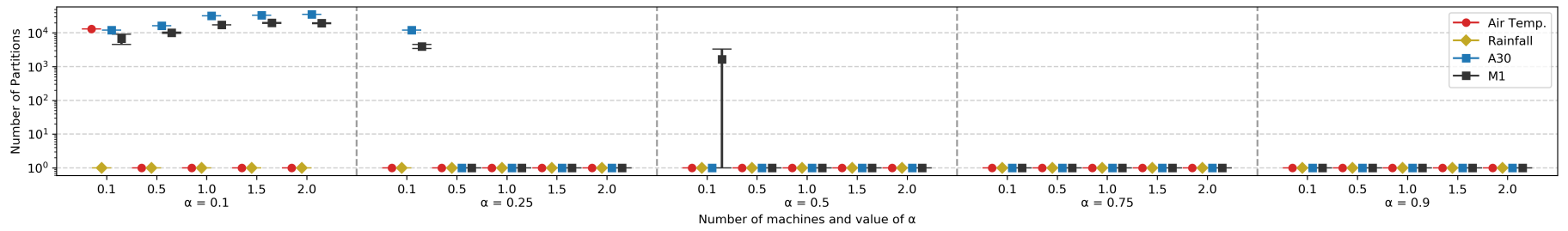
Table H.4: Spatial perturbations: variance of the raw clean datasets, raw perturbed datasets and reduced perturbed datasets. Results are shown for DTR modelling on partitions. Values of  $\alpha$  close to 0 indicate high storage used for minimised error, and values of  $\alpha$  close to 1 indicate minimised storage but higher error.

$\alpha$	SD.	Clean	A30 Pert.	Recon.	Clean	M1 Pert.	Recon.	Air Temperature			Clean	Rainfall Pert.	Recon.
0.1	1.5	21130	21130	19844	235927	235927	227610	11.1	11.1	8.79	0.272	0.272	0.171
	350	21130	21130	19839	235927	235927	229152	11.1	11.1	8.8	0.272	0.272	0.164
	700	21130	21130	19820	235927	235927	223419	11.1	11.1	8.79	0.272	0.272	0.164
	1029	21130	21130	19802	235927	235927	223762	11.1	11.1	8.79	0.272	0.272	0.172
	2000	21130	21130	19778	235927	235927	223892	11.1	11.1	8.77	0.272	0.272	0.165
0.25	1.5	21130	21130	12956	235927	235927	219532	11.1	11.1	8.51	0.272	0.272	0.057
	350	21130	21130	6132	235927	235927	218766	11.1	11.1	8.51	0.272	0.272	0.057
	700	21130	21130	12431	235927	235927	218846	11.1	11.1	8.51	0.272	0.272	0.057
	1029	21130	21130	12689	235927	235927	219061	11.1	11.1	8.51	0.272	0.272	0.058
	2000	21130	21130	12664	235927	235927	219287	11.1	11.1	8.49	0.272	0.272	0.058
0.5	1.5	21130	21130	5281	235927	235927	219143	11.1	11.1	8.22	0.272	0.272	0.031
	350	21130	21130	4756	235927	235927	218655	11.1	11.1	8.23	0.272	0.272	0.031
	700	21130	21130	4634	235927	235927	219023	11.1	11.1	8.22	0.272	0.272	0.035
	1029	21130	21130	4826	235927	235927	218933	11.1	11.1	8.22	0.272	0.272	0.031
	2000	21130	21130	4829	235927	235927	148444	11.1	11.1	8.21	0.272	0.272	0.031
0.75	1.5	21130	21130	3852	235927	235927	76839	11.1	11.1	7.46	0.272	0.272	0.01
	350	21130	21130	3809	235927	235927	57936	11.1	11.1	7.47	0.272	0.272	0.01
	700	21130	21130	4330	235927	235927	74125	11.1	11.1	7.46	0.272	0.272	0.01
	1029	21130	21130	3555	235927	235927	74644	11.1	11.1	7.45	0.272	0.272	0.01
	2000	21130	21130	3747	235927	235927	74584	11.1	11.1	7.43	0.272	0.272	0.01
0.9	1.5	21130	21130	3852	235927	235927	72341	11.1	11.1	6.95	0.272	0.272	0.005
	350	21130	21130	3809	235927	235927	62336	11.1	11.1	6.96	0.272	0.272	0.005
	700	21130	21130	3422	235927	235927	72559	11.1	11.1	6.96	0.272	0.272	0.005
	1029	21130	21130	3290	235927	235927	71573	11.1	11.1	6.95	0.272	0.272	0.005
	2000	21130	21130	2447	235927	235927	72260	11.1	11.1	6.93	0.272	0.272	0.005

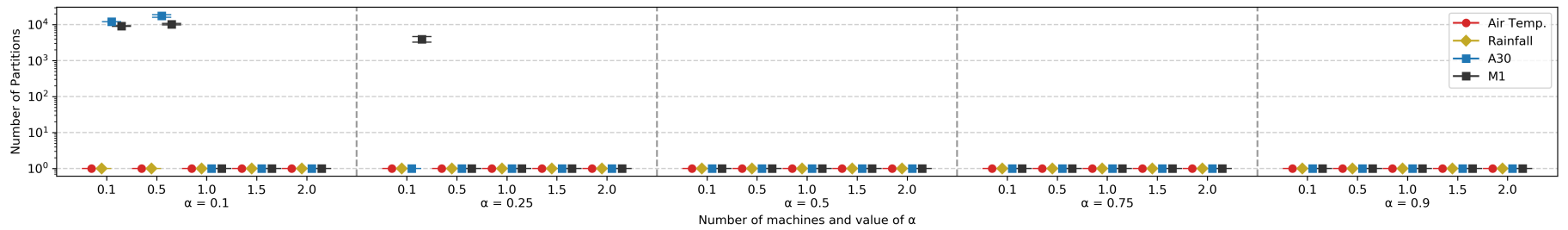




(a) PLR-P modelling

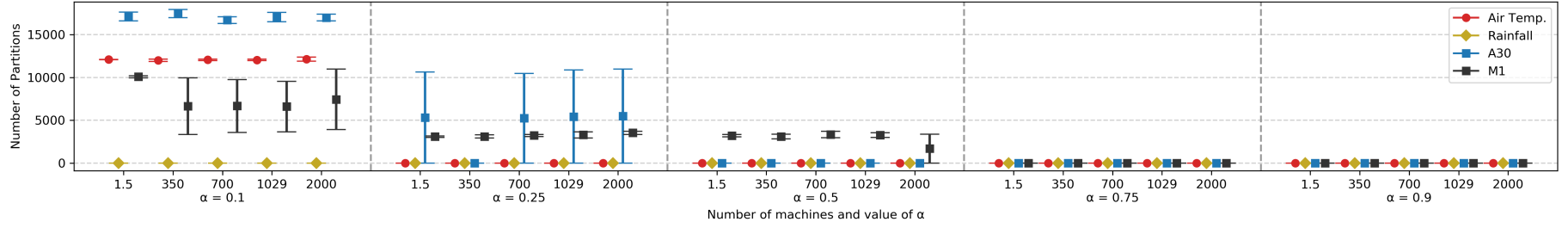


(b) DCT-P modelling

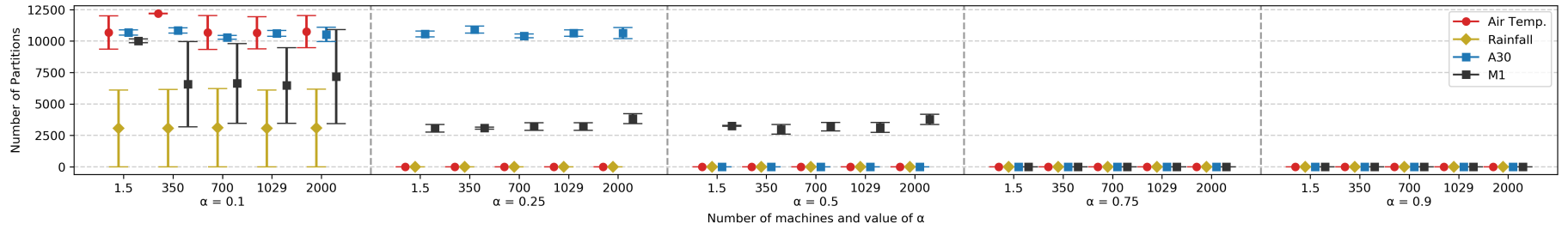


(c) DTR-P modelling

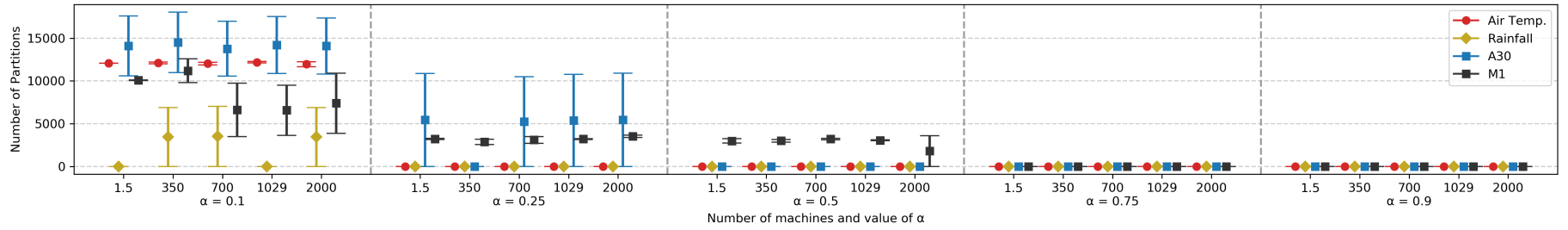
Figure H.1: Number of partitions in the reduced form of the datasets with noise.



(a) PLR-P modelling

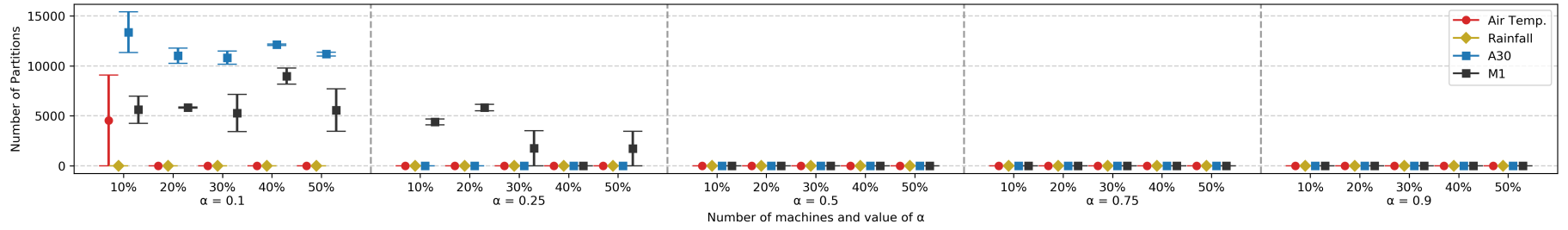


(b) DCT-P modelling

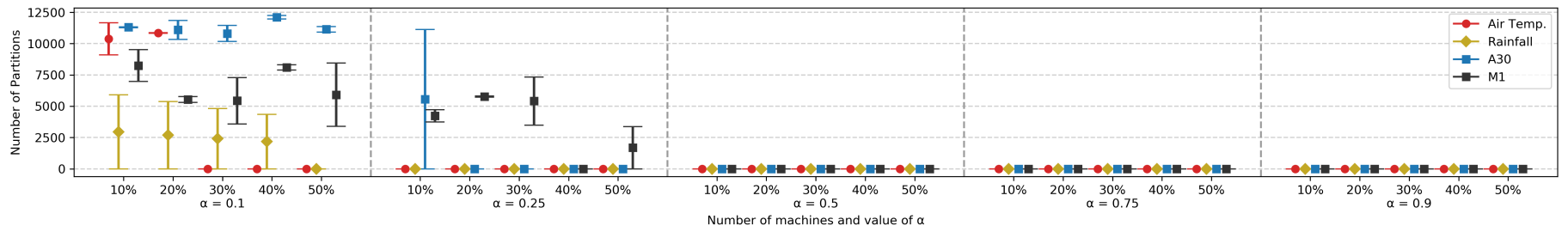


(c) DTR-P modelling

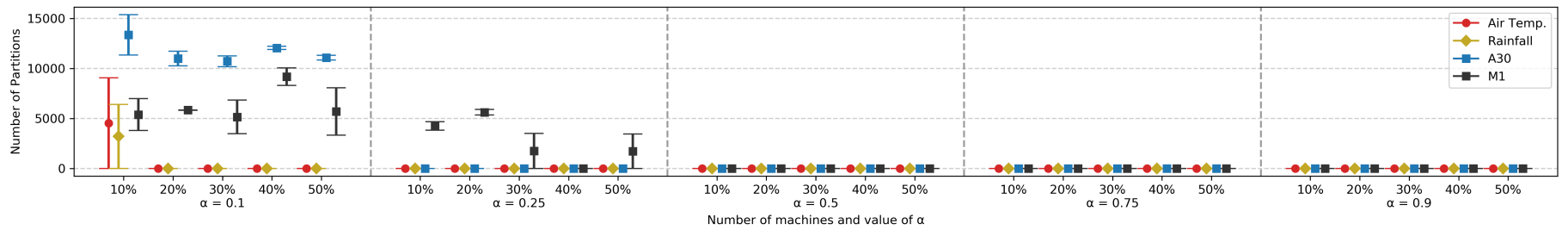
Figure H.2: Number of partitions in the reduced form of the datasets with spatial error.



(a) PLR-P modelling

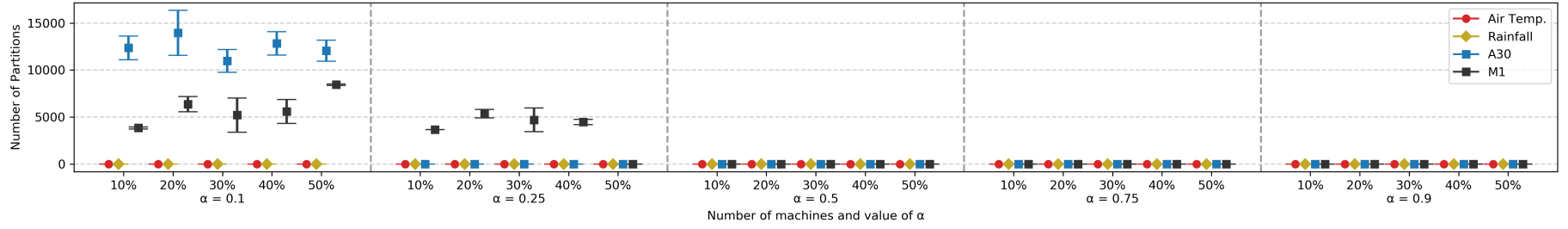


(b) DCT-P modelling

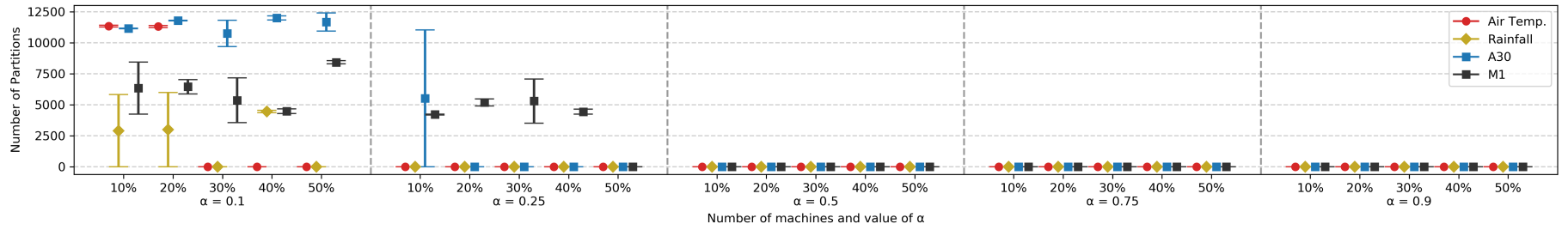


(c) DTR-P modelling

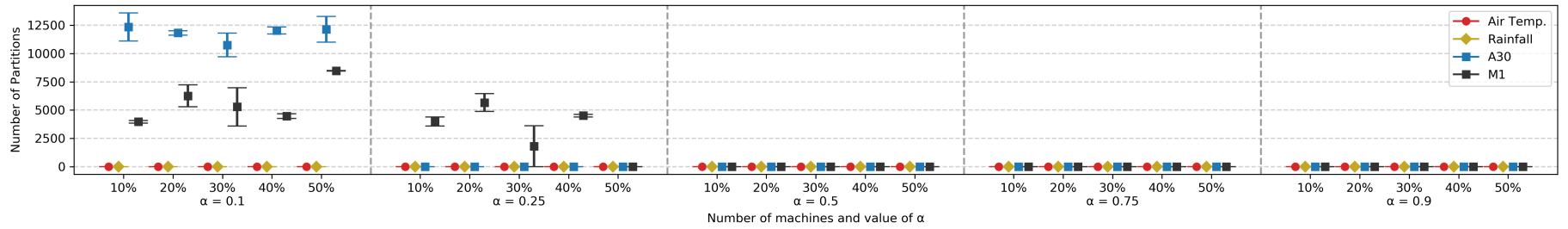
Figure H.3: Number of partitions in the reduced form of the datasets with instances missing at random.



(a) PLR-P modelling



(b) DCT-P modelling

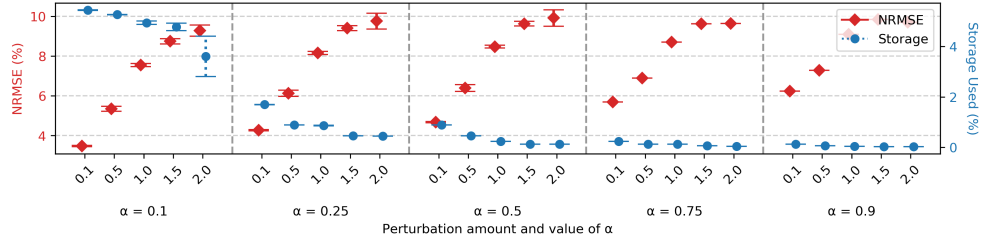


(c) DTR-P modelling

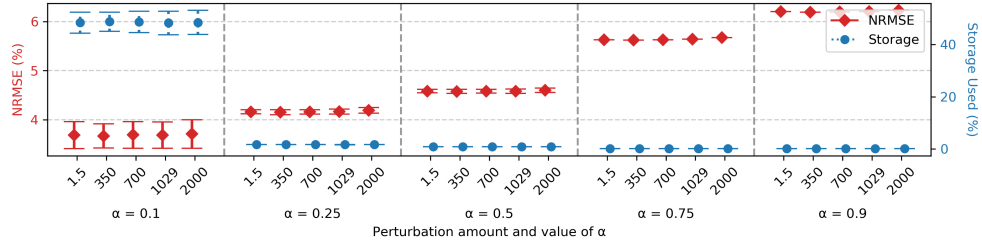
Figure H.4: Number of partitions in the reduced form of the datasets with instances missing not at random.

### **H.3 Effect on Reconstruction Error and Storage**

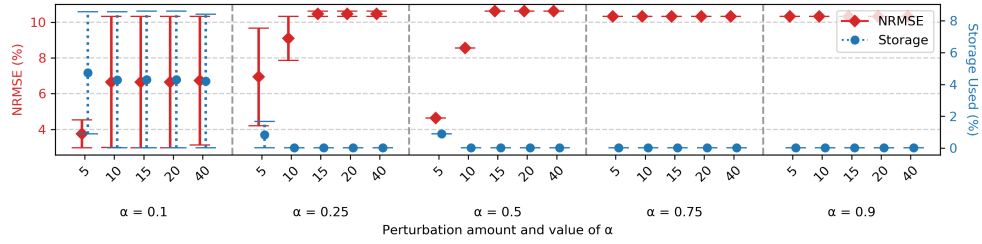
These results show the storage used and reconstruction NRMSE incurred by the reduced datasets as the amount of error, noise and missing data increases. Results are shown for DTR-P modelling, and results for the other modelling techniques can be found in the online repository [120].



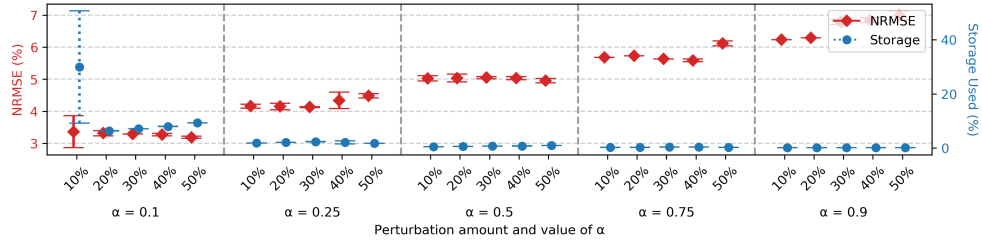
(a) Noise (Air Temperature, DTR-P)



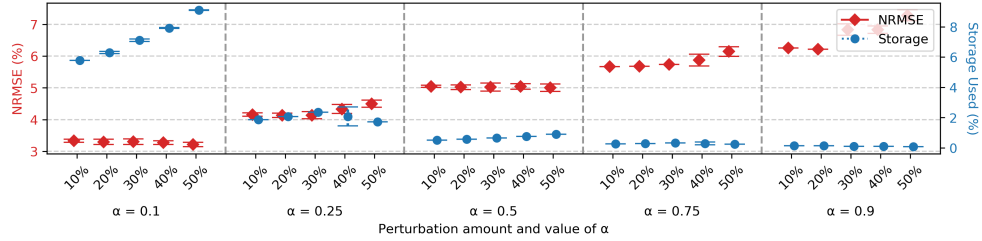
(b) Spatial Error (Air Temperature, DTR-P)



(c) Temporal Error (Air Temperature, DTR-P)

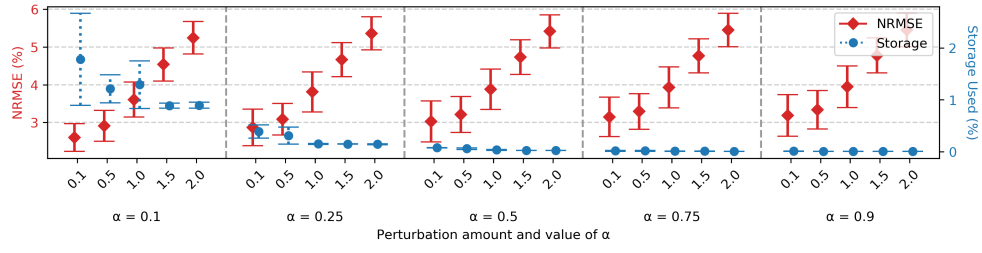


(d) Missing at Random (Air Temperature, DTR-P)

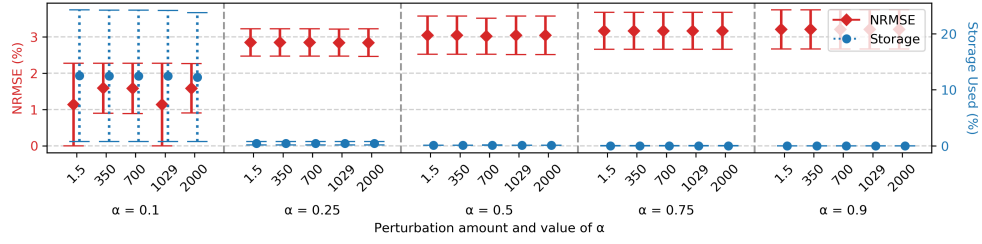


(e) Missing Not at Random (Air Temperature, DTR-P)

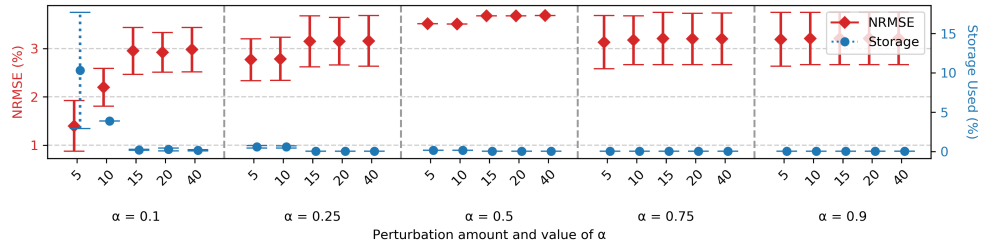
Figure H.5: Storage used and reconstruction NRMSE incurred for air temperature dataset (DTR-P modelling).



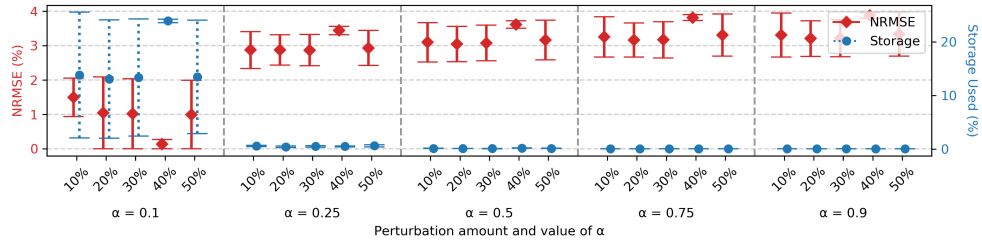
(a) Noise (Rainfall, DTR-P)



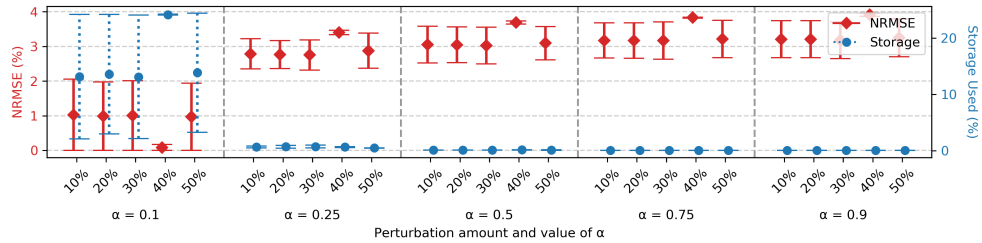
(b) Spatial Error (Rainfall, DTR-P)



(c) Temporal Error (Rainfall, DTR-P)

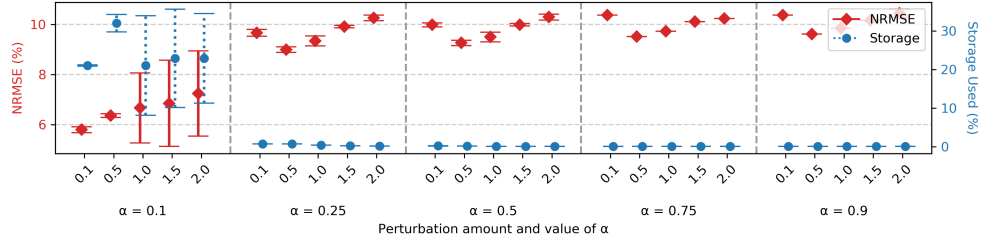


(d) Missing at Random (Rainfall, DTR-P)

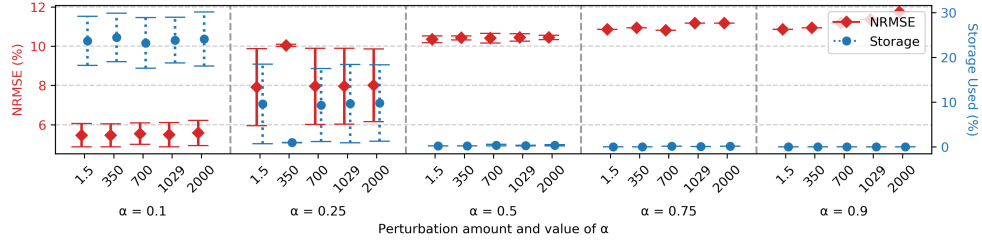


(e) Missing Not at Random (Rainfall, DTR-P)

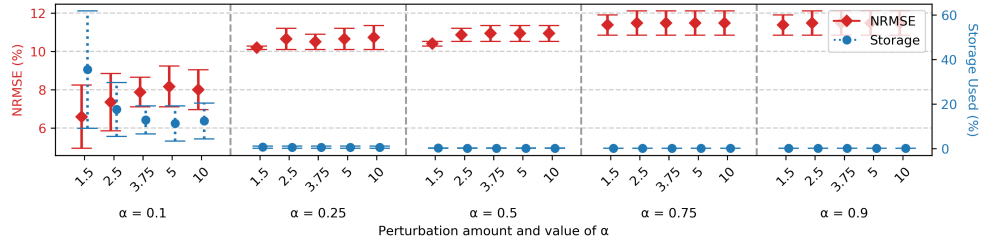
Figure H.6: Storage used and reconstruction NRMSE incurred for rainfall dataset (DTR-P modelling).



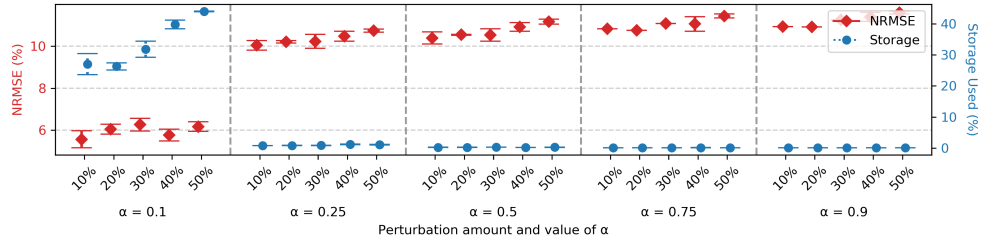
(a) Noise (A30 Traffic, DTR-P)



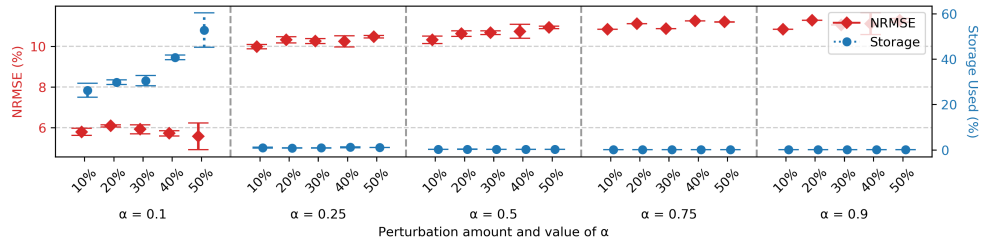
(b) Spatial Error (A30 Traffic, DTR-P)



(c) Temporal Error (A30 Traffic, DTR-P)



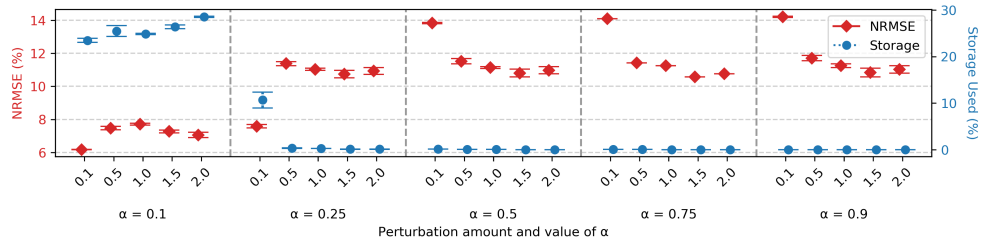
(d) Missing at Random (A30 Traffic, DTR-P)



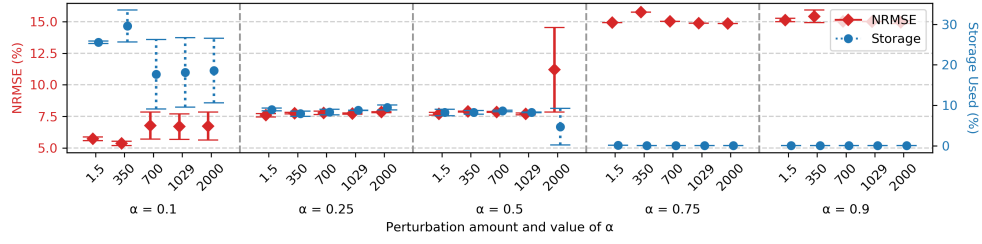
(e) Missing Not at Random (A30 Traffic, DTR-P)

Figure H.7: Storage used and reconstruction NRMSE incurred for A30 traffic dataset (DTR-P modelling).

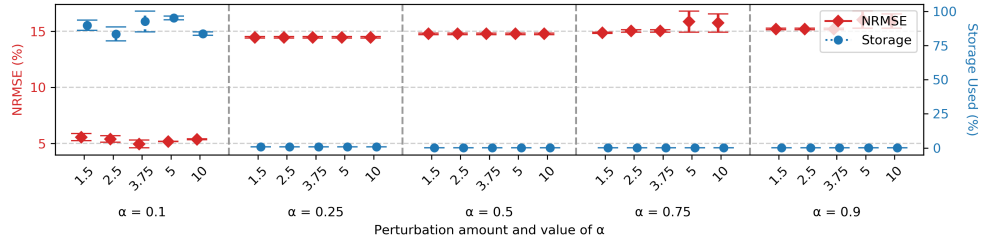




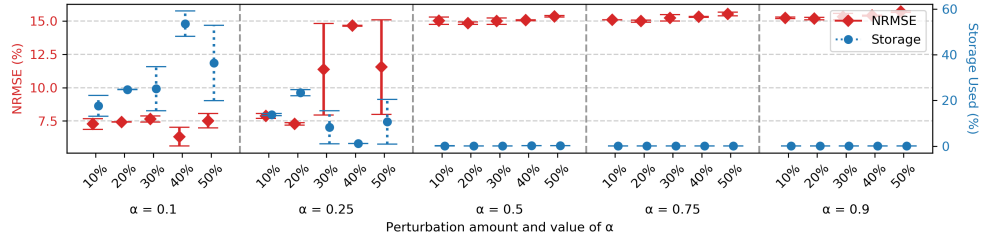
(a) Noise (M1 Traffic, DTR-P)



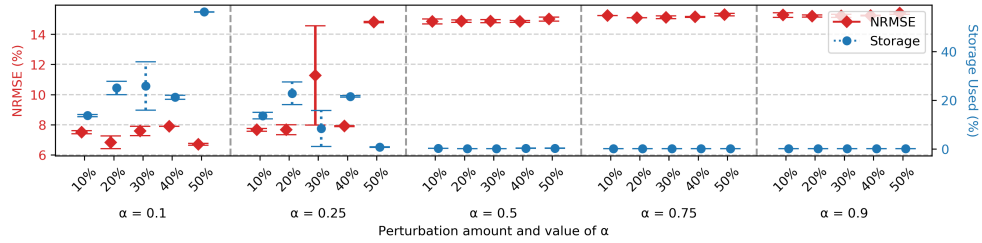
(b) Spatial Error (M1 Traffic, DTR-P)



(c) Temporal Error (M1 Traffic, DTR-P)



(d) Missing at Random (M1 Traffic, DTR-P)

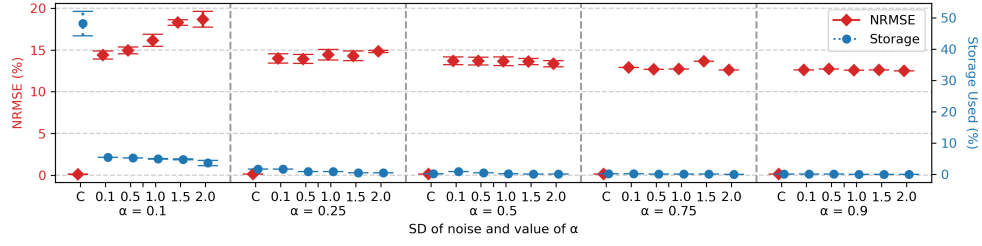


(e) Missing Not at Random (M1 Traffic, DTR-P)

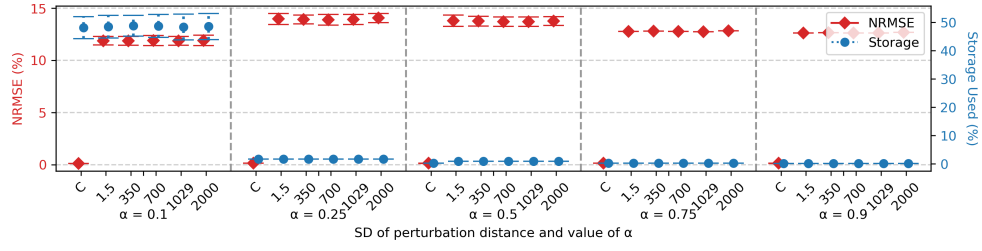
Figure H.8: Storage used and reconstruction NRMSE incurred for M1 traffic dataset (DTR-P modelling).

## H.4 Effect on Imputation Error

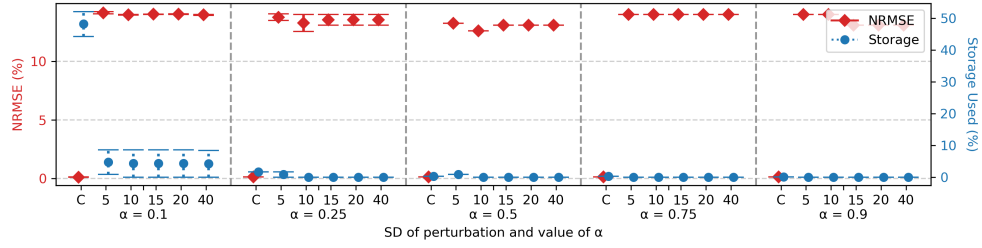
These results show the storage used and imputation NRMSE incurred by the reduced datasets as the amount of error, noise and missing data increases. Results are shown for DTR-P modelling, and results for the other modelling techniques can be found in the online repository [120].



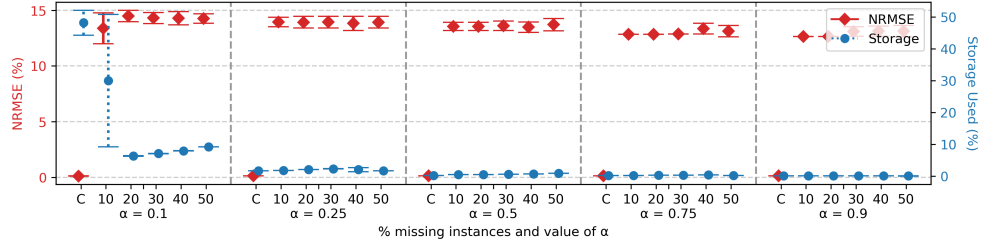
(a) Noise (Air Temperature, DTR-P)



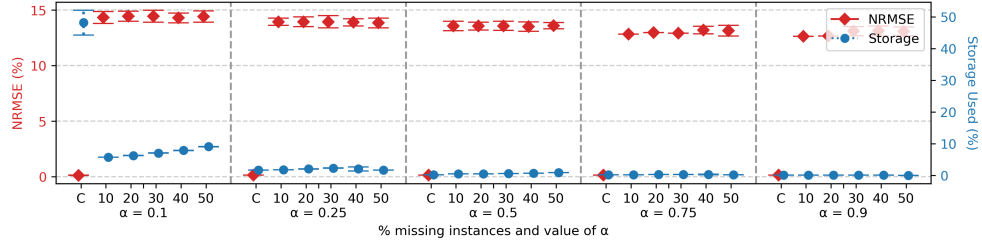
(b) Spatial Error (Air Temperature, DTR-P)



(c) Temporal Error (Air Temperature, DTR-P)

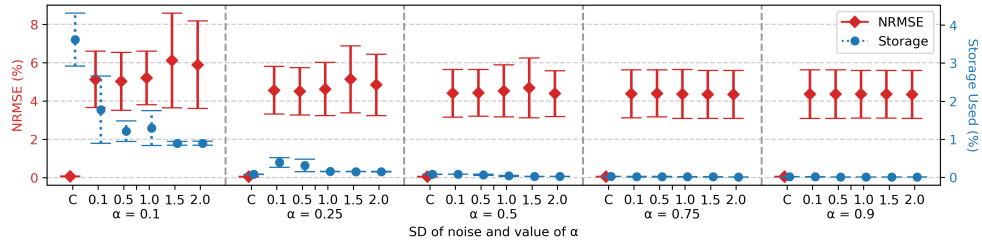


(d) Missing at Random (Air Temperature, DTR-P)

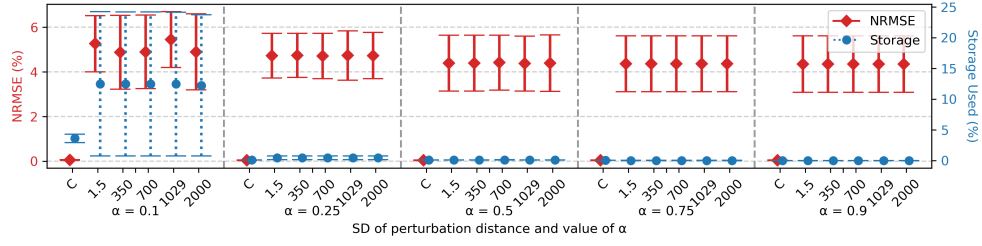


(e) Missing Not at Random (Air Temperature, DTR-P)

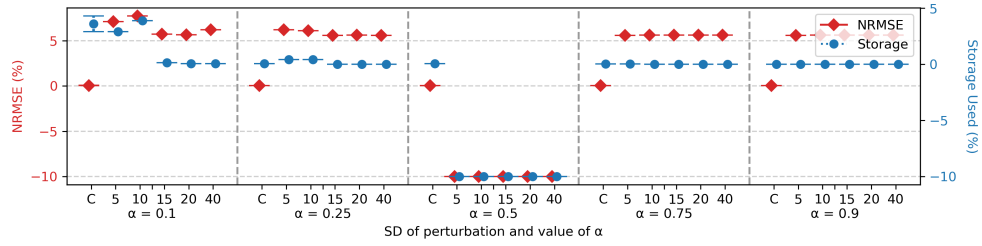
Figure H.9: Storage used and imputation NRMSE incurred for air temperature dataset (DTR-P modelling).



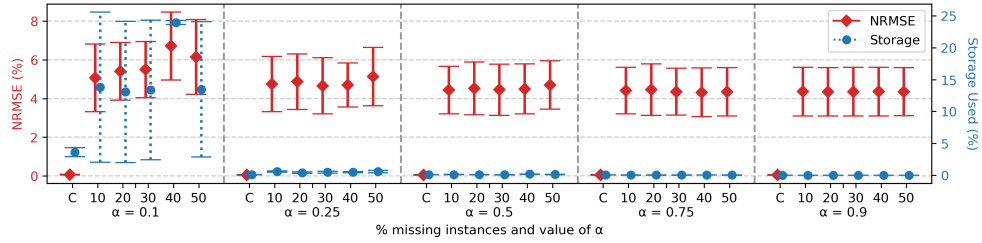
(a) Noise (Rainfall, DTR-P)



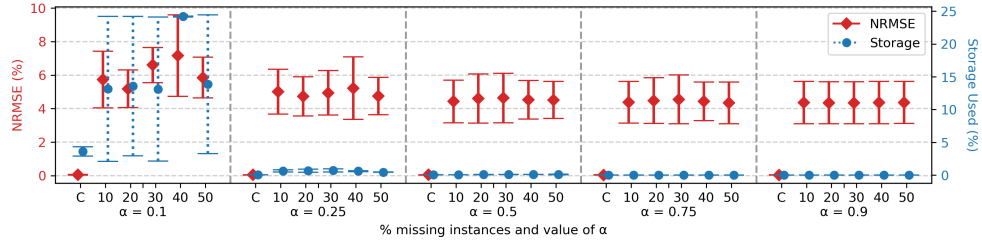
(b) Spatial Error (Rainfall, DTR-P)



(c) Temporal Error (Rainfall, DTR-P)

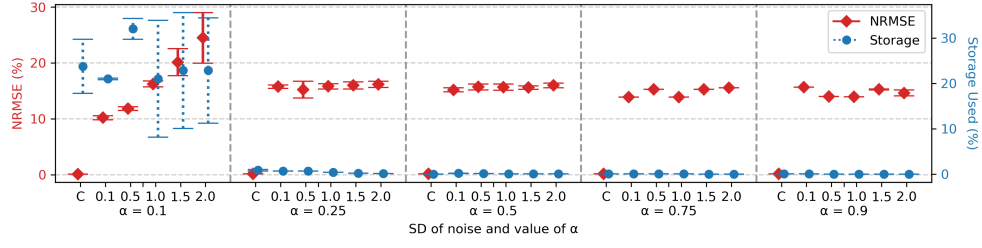


(d) Missing at Random (Rainfall, DTR-P)

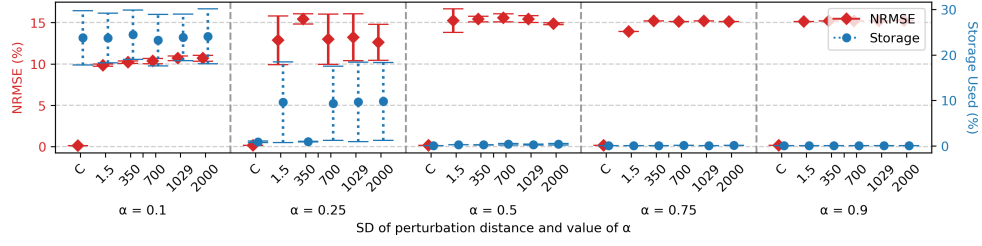


(e) Missing Not at Random (Rainfall, DTR-P)

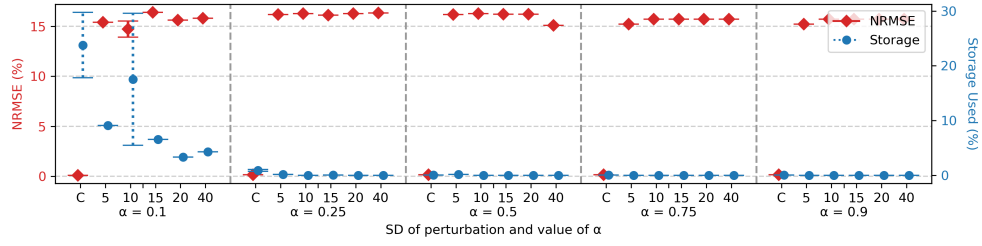
Figure H.10: Storage used and imputation NRMSE incurred for rainfall dataset (DTR-P modelling).



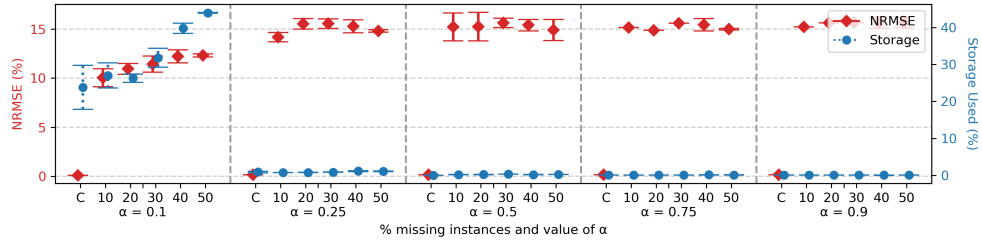
(a) Noise (A30 Traffic, DTR-P)



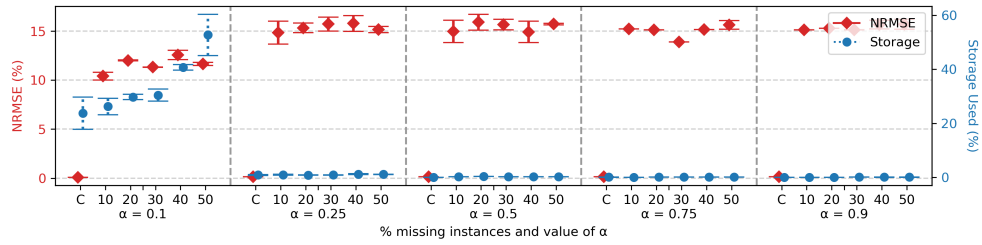
(b) Spatial Error (A30 Traffic, DTR-P)



(c) Temporal Error (A30 Traffic, DTR-P)

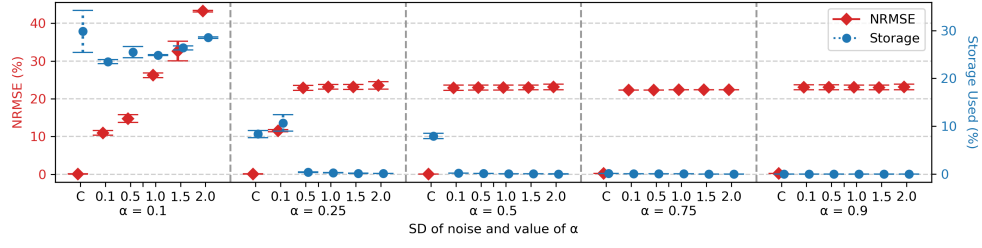


(d) Missing at Random (A30 Traffic, DTR-P)

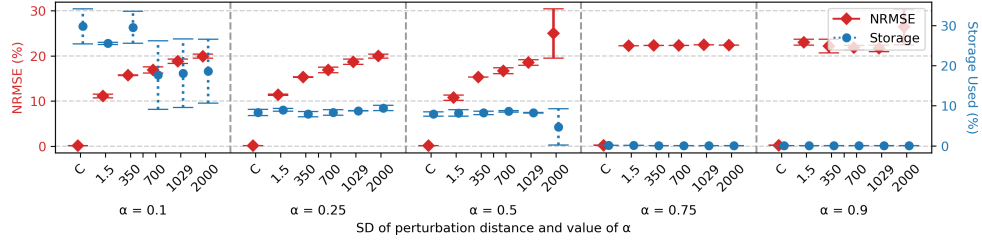


(e) Missing Not at Random (A30 Traffic, DTR-P)

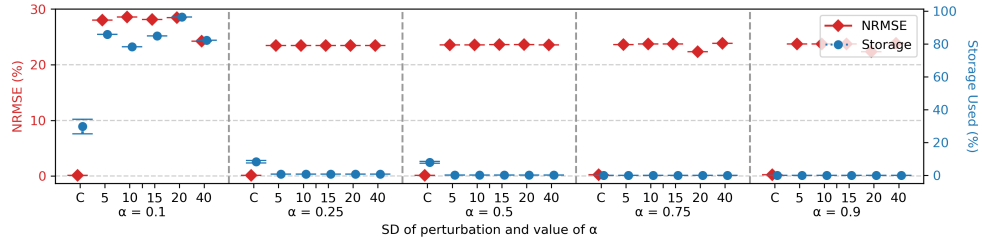
Figure H.11: Storage used and imputation NRMSE incurred for A30 traffic dataset (DTR-P modelling).



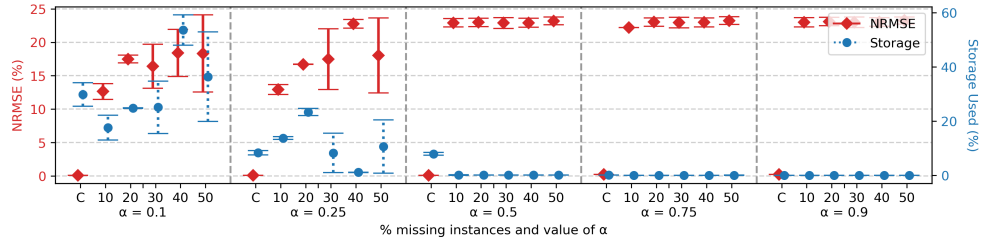
(a) Noise (M1 Traffic, DTR-P)



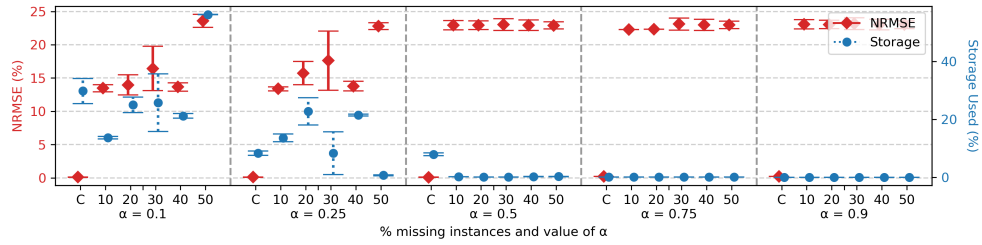
(b) Spatial Error (M1 Traffic, DTR-P)



(c) Temporal Error (M1 Traffic, DTR-P)



(d) Missing at Random (M1 Traffic, DTR-P)



(e) Missing Not at Random (M1 Traffic, DTR-P)

Figure H.12: Storage used and imputation NRMSE incurred for M1 traffic dataset (DTR-P modelling).

# Bibliography

- [1] Highways England Pavement Management System Road Condition Survey. Technical report, 2019.
- [2] Charu C Aggarwal. Spatial Outlier Detection. In *Outlier Analysis*, pages 345–368. Springer International Publishing, Cham, 2017.
- [3] David Aha, Dennis Kibler, and Marc Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, January 1991.
- [4] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. Multi-level Techniques for Compression and Reduction of Scientific Data—The Univariate Case. *Computing and Visualization in Science*, 19(5):65–76, 2018.
- [5] Mohammad Akbari, Farhad Samadzadegan, and Robert Weibel. A Generic Regional Spatio-Temporal Co-occurrence Pattern Mining Model: A Case Study for Air Pollution. *Journal of Geographical Systems*, 17(3): 249–274, July 2015.
- [6] Alaa M Al-Abadi. Modeling of Groundwater Productivity in Northeastern Wasit Governorate, Iraq using Frequency Ratio and Shannon’s Entropy Models. *Applied Water Science*, 7(2):699–716, 2017.
- [7] Louai Alarabi, Mohamed F Mokbel, and Mashaal Musleh. ST-Hadoop: A MapReduce Framework for Spatio-Temporal Data. *Geoinformatica*, 22(4):785–813, October 2018.
- [8] James F Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.
- [9] Hussein Almuallim and Thomas G Dietterich. Learning with Many Irrelevant Features. In *9th National Conference on Artificial Intelligence*, pages 547–552. AAAI Press, 1991.
- [10] Gennady Andrienko and Natalia Andrienko. Interactive Cluster Analysis of Diverse Types of Spatiotemporal Data. *ACM SIGKDD Explorations Newsletter*, 11(2):19–28, 2010.

- [11] J C K Arulswamy and S L Pallickara. Columbus: Enabling Scalable Scientific Workflows for Fast Evolving Spatio-Temporal Sensor Data. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 9–18, 2017.
- [12] A Aswi, S M Cramb, P Moraga, and K Mengersen. Bayesian Spatial and Spatio-Temporal Approaches to Modelling Dengue Fever: A Systematic Review. *Epidemiology and infection*, 147:1–14, October 2018.
- [13] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. Spatio-Temporal Data Mining: A Survey of Problems and Methods. *ACM Computing Surveys*, 51(4), August 2018.
- [14] Franz Aurenhammer. Voronoi Diagrams—a Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.
- [15] Mukund Balasubramanian, Eric L Schwartz, Joshua B Tenenbaum, Vin de Silva, and John C Langford. The Isomap Algorithm and Topological Stability. *Science*, 295(5552):7–7, 2002.
- [16] MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity Clustering: Hierarchical Clustering at Scale. In *Advances in Neural Information Processing Systems*, pages 6864–6874, 2017.
- [17] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, September 1975.
- [18] Dimitris Berberidis and Georgios B Giannakis. Data Sketching for Large-Scale Kalman Filtering. *IEEE Transactions on Signal Processing*, 65(14):3688–3701, July 2017.
- [19] Maria Bermudez-Edo, Payam Barnaghi, and Klaus Moessner. Analysing Real World Data Streams with Spatio-Temporal Correlations: Entropy vs. Pearson Correlation. *Automation in Construction*, 88:87–100, 2018.
- [20] Derya Birant and Alp Kut. ST-DBSCAN: An algorithm for Clustering Spatial–Temporal Data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.
- [21] D Birvinskas, V Jusas, I Martisius, and R Damasevicius. EEG Dataset Reduction and Feature Extraction Using Discrete Cosine Transform. In



2012 Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation, pages 199–204, November 2012.

- [22] Burton H Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, July 1970.
- [23] Neeraj Bokde, Marcus W Beck, Francisco Martínez Álvarez, and Kishore Kulat. A Novel Imputation Methodology for Time Series Based on Pattern Sequence Forecasting. *Pattern Recognition Letters*, 116:88–96, 2018.
- [24] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and Regression Trees*. CRC press, 1984.
- [25] A Z Broder. On the Resemblance and Containment of Documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29, June 1997.
- [26] Katarzyna Cegielska, Anita Kukulska-Koziel, Tomasz Salata, Piotr Piotrowski, and Marta Szylar. Shannon Entropy as a Peri-Urban Landscape Metric: Concentration of Anthropogenic Land Cover Element. *Journal of Spatial Science*, 64(3):469–489, 2019.
- [27] Chin-Liang Chang. Finding Prototypes For Nearest Neighbor Classifiers. *IEEE Transactions on Computers*, C-23(11):1179–1184, November 1974.
- [28] Siguang Chen, Jincheng Liu, Kun Wang, and Meng Wu. A Hierarchical Adaptive Spatio-Temporal Data Compression Scheme for Wireless Sensor Networks. *Wireless Networks*, 25(1):429–438, January 2019.
- [29] Xinyu Chen, Zhaocheng He, and Lijun Sun. A Bayesian Tensor Decomposition Approach for Spatiotemporal Traffic Data Imputation. *Transportation Research Part C: Emerging Technologies*, 98:73–84, 2019.
- [30] Ching-Hsue Cheng, Chia-Pang Chan, and Yu-Jheng Sheu. A Novel Purity-Based k Nearest Neighbors Imputation Method and its Application in Financial Distress Prediction. *Engineering Applications of Artificial Intelligence*, 81:283–299, 2019.
- [31] Michael Chipeta, Dianne Terlouw, Kamija Phiri, and Peter Diggle. Inhibitory Geostatistical Designs for Spatial Prediction Taking Account of Uncertain Covariance Structure. *Environmetrics*, 28(1):e2425, 2017.
- [32] Cora Hoi-Shan Chong. *Comparison of Spatial Data Types for Urban Sprawl Analysis Using Shannon’s Entropy*. PhD thesis, University of Southern California, 2017.

- [33] AB Arockia Christopher and S Appavu Balamurugan. Feature Selection Techniques for Prediction of Warning Level in Aircraft Accidents. In *2013 International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–6. IEEE, 2013.
- [34] Zun Liang Chuan, Sayang Mohd Deni, Soo-Fen Fam, and Noriszura Ismail. The Effectiveness of a Probabilistic Principal Component Analysis Model and Expectation Maximisation Algorithm in Treating Missing Daily Rainfall Data. *Asia-Pacific Journal of Atmospheric Sciences*, 56(1): 119–129, 2020.
- [35] Christophe Claramunt. Towards a Spatio-Temporal Form of Entropy. In *International Conference on Conceptual Modeling*, pages 221–230. Springer, 2012.
- [36] Graham Cormode and Shan Muthukrishnan. An Improved Data Stream Summary: the Count-Min Sketch and its Applications. *Journal of Algorithms*, 55(1):58–75, 2005.
- [37] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.
- [38] Noel Cressie and Christopher K Wikle. *Statistics for Spatio-Temporal Data*. John Wiley & Sons, 2015.
- [39] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1): 107–113, January 2008.
- [40] Eric Delmelle, Coline Dony, Irene Casas, Meijuan Jia, and Wenwu Tang. Visualizing the Impact of Space-Time Uncertainties on Dengue Fever Patterns. *International Journal of Geographical Information Science*, 28(5):1107–1127, 2014.
- [41] Eric M Delmelle. Spatial Sampling. In Manfred M Fischer and Peter Nijkamp, editors, *Handbook of Regional Science*, pages 1385–1399. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
- [42] Urvska Demvsar, Paul Harris, Chris Brunsdon, A Stewart Fotheringham, and Sean McLoone. Principal Component Analysis on Spatial Data: An Overview. *Annals of the Association of American Geographers*, 103(1): 106–128, 2013.
- [43] Urvska Demvsar, Paul Harris, Chris Brunsdon, A Stewart Fotheringham, and Sean McLoone. Principal Component Analysis on Spatial Data: An

- Overview. *Annals of the Association of American Geographers*, 103(1): 106–128, 2013.
- [44] M R Desjardins, A Hohl, and E M Delmelle. Rapid Surveillance of COVID-19 in the United States using a Prospective Space-Time Scan Statistic: Detecting and Evaluating Emerging Clusters. *Applied Geography*, 118: 102202, 2020.
  - [45] P Deutsch. DEFLATE Compressed Data Format Specification version 1.3. *RFC*, 1951:1–17, 1996.
  - [46] Y Ding, Y Li, K Deng, H Tan, M Yuan, and L M Ni. Detecting and Analyzing Urban Regions with High Impact of Weather Change on Transport. *IEEE Transactions on Big Data*, 3(2):126–139, June 2017.
  - [47] Highways England. Highways England Network Journey Time and Traffic Flow Data. Technical report, 2018.
  - [48] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and others. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.
  - [49] European Organization for the Safety of Air Navigation (EUROCONTROL), Institute of Geodesy, and Navigation IfEN. WGS 84 Implementation Manual. Technical report, 1998.
  - [50] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the Analysis of a Near-Optimal Cardinality Estimation Algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156, 2007.
  - [51] Deepak Ganesan, Sylvia Ratnasamy, Hanbiao Wang, and Deborah Estrin. Coping with Irregular Spatio-Temporal Sampling in Sensor Networks. *ACM SIGCOMM Computer Communication Review*, 34(1):125–130, 2004.
  - [52] Nan Gao, Hao Xue, Wei Shao, Sichen Zhao, Kyle Kai Qin, Arian Prabowo, Mohammad Saiedur Rahaman, and Flora D Salim. Generative Adversarial Networks for Spatio-temporal Data: A Survey. Technical report, 2020.
  - [53] Fernando Garcia-Menendez, Yongtao Hu, and Mehmet T Odman. Simulating Smoke Transport from Wildland Fires with a Regional-Scale Air Quality Model: Sensitivity to Spatiotemporal Allocation of Fire Emissions. *Science of the Total Environment*, 493:544–553, 2014.

- [54] Irene Gargantini. An Effective Way to Represent Quadtrees. *Communications of the ACM*, 25(12):905–910, December 1982.
- [55] GeoBuiz. Geospatial Industry Outlook and Readiness Index: 2019 Edition. March 2019.
- [56] Shlomo Geva and Joaquin Sitte. Adaptive Nearest Neighbor Pattern Classification. *IEEE Transactions on Neural Networks*, 2(2):318–322, March 1991.
- [57] Yikai Gong, Paul Rimba, and Richard Sinnott. A Big Data Architecture for Near Real-Time Traffic Analytics. In *10th International Conference on Utility and Cloud Computing*, pages 157–162, New York, NY, USA, 2017. ACM.
- [58] Daniel A Griffith, Yongwan Chun, and Monghyeon Lee. Locational Error Impacts on Local Spatial Autocorrelation Indices: A Syracuse Soil Sample Pb-Level Data Case Study. *Spatial Accuracy Assessment in Natural Resources and Environmental Sciences, Montpellier, France*, pages 5–8, 2016.
- [59] M Gupta, J Gao, C C Aggarwal, and J Han. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, September 2014.
- [60] Yaobin He, Haoyu Tan, Wuman Luo, Shengzhong Feng, and Jianping Fan. MR-DBSCAN: a Scalable MapReduce-Based DBSCAN Algorithm for Heavily Skewed Data. *Frontiers of Computer Science*, 8(1):83–99, 2014.
- [61] Zhenwen He, Gang Liu, Xiaogang Ma, and Qiyu Chen. GeoBeam: A Distributed Computing Framework for Spatial Data. *Computers & Geosciences*, 131:15–22, 2019.
- [62] William Hendrix, Md Mostofa Ali Patwary, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. Parallel Hierarchical Clustering on Shared Memory Platforms. In *2012 19th International Conference on High Performance Computing*, pages 1–9. IEEE, 2012.
- [63] Liang Heng, Grace Xingxin Gao, Todd Walter, and Per Enge. Statistical Characterization of GPS Signal-in-Space Errors. In *International Technical Meeting of the Institute of Navigation (ION ITM 2011), San Diego, CA*, pages 312–319. Citeseer, 2011.
- [64] Bo Huang, Bo Wu, and Michael Barry. Geographically and Temporally Weighted Regression for Modeling Spatio-Temporal Variation in House

- Prices. *International Journal of Geographical Information Science*, 24 (3):383–401, 2010.
- [65] Sitharama S Iyengar, Kianoosh G Boroojeni, and N Balakrishnan. *Mathematical Theories of Distributed Sensor Networks*. Springer, 2014.
  - [66] Andreas Janecek, Wilfried Gansterer, Michael Demel, and Gerhard Ecker. On the Relationship Between Feature Selection and Classification Accuracy. In Yvan Saeys, Huan Liu, Iñaki Inza, Louis Wehenkel, and Yves Van de Pee, editors, *Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008*, pages 90–105. PMLR, September 2008.
  - [67] Xiaowei Jia, Ankush Khandelwal, Guruprasad Nayak, James Gerber, Kimberly Carlson, Paul West, and Vipin Kumar. Incremental dual-Memory LSTM in Land Cover Prediction. In *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 867–876, 2017.
  - [68] C Jin, R Liu, Z Chen, W Hendrix, A Agrawal, and A Choudhary. A Scalable Hierarchical Clustering Algorithm Using Spark. In *First International Conference on Big Data Computing Service and Applications*, pages 418–426. IEEE, March 2015.
  - [69] Chen Jin, Md Mostofa Ali Patwary, Ankit Agrawal, William Hendrix, Wei-keng Liao, and Alok Choudhary. DiSC: A Distributed Single-Linkage Hierarchical Clustering Algorithm using MapReduce. *Work*, 23:27, 2013.
  - [70] Kenji Kira and Larry A Rendell. The Feature Selection Problem: Traditional Methods and a New Algorithm. In *Proceedings of 10th National Conference on Artificial Intelligence*, pages 129–134, 1992.
  - [71] Christopher R Knittel, Douglas L Miller, and Nicholas J Sanders. Caution, Drivers! Children Present: Traffic, Pollution, and Infant Health. *The Review of Economics and Statistics*, 98(2):350–366, 2016.
  - [72] Teuvo Kohonen, editor. *Self-Organizing Maps*. Springer-Verlag, 3rd edition, 2001.
  - [73] Xiangjie Kong, Menglin Li, Jianxin Li, Kaiqi Tian, Xiping Hu, and Feng Xia. CoPFun: An Urban Co-Occurrence Pattern Mining Scheme based on Regional Function Discovery. *World Wide Web*, 22(3):1029–1054, May 2019.
  - [74] Martin Kulldorff. A Spatial Scan Statistic. *Communications in Statistics - Theory and Methods*, 26(6):1481–1496, 1997.

- [75] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Seung-Hoe Ku, C S Chang, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F Samatova. ISABELA for Effective in situ Compression of Scientific Data. *Concurrency and Computation: Practice and Experience*, 25(4):524–540, 2013.
- [76] Nhu D Le and James V Zidek. *Statistical Analysis of Environmental Space-Time Processes*. Springer Science & Business Media, 2006.
- [77] Laure Lebecherel, Vazken Andréassian, and Charles Perrin. On Evaluating the Robustness of Spatial-Proximity-Based Regionalization Methods. *Journal of Hydrology*, 539:196–203, 2016.
- [78] Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. Improving Statistical Similarity Based Data Reduction for Non-Stationary Data. In *29th International Conference on Scientific and Statistical Database Management*, pages 37:1–37:6. ACM, 2017.
- [79] Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. IDEALEM: Statistical Similarity Based Data Reduction. Technical report, 2019.
- [80] Krisada Lekdee and L Ingsrisawang. Generalized Linear Mixed Models with Spatial Random Effects for Spatio-Temporal Data: An Application to Dengue Fever Mapping. *Journal of Mathematics and Statistics*, 9: 137–143, 2013.
- [81] M Lévesque and D Tipper. A Survey of Clock Synchronization Over Packet-Switched Networks. *IEEE Communications Surveys Tutorials*, 18(4):2926–2947, 2016.
- [82] Jundong Li, Kewei Cheng, Suhan Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature Selection: A Data Perspective. *ACM Computing Surveys*, 50(6):94:1–94:45, December 2017.
- [83] Xin Li, Xingtao Ou, Zhi Li, Henglui Wei, Wei Zhou, and Zhemin Duan. On-Line Temperature Estimation for Noisy Thermal Sensors Using a Smoothing Filter-Based Kalman Predictor. *Sensors*, 18(2):433, 2018.
- [84] Qin Liu, Ran Chen, Hongming Zhu, and Hongfei Fan. Research and Comparison of Data Dimensionality Reduction Algorithms. In *International Conference on Bioinformatics and Computational Intelligence*, pages 49–54. ACM, 2017.
- [85] Yi Lu, Anil Shanbhag, Alekh Jindal, and Samuel Madden. AdaptDB: Adaptive Partitioning for Distributed Joins. *VLDB Endowment*, 10(5): 589–600, 2017.

- [86] Z Lv, X Li, H Lv, and W Xiu. BIM Big Data Storage in WebVRGIS. *IEEE Transactions on Industrial Informatics*, 16(4):2566–2573, April 2020.
- [87] Nicholas Malizia. Inaccuracy, Uncertainty and the Space-Time Permutation Scan Statistic. *PloS One*, 8(2):e52034, 2013.
- [88] Aleix M Martinez and Avinash C Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, February 2001.
- [89] Syrine Ben Meskina. On the Effect of Data Reduction on Classification Accuracy. In *3rd International Conference on Information Technology and e-Services (ICITeS)*, pages 1–7. IEEE, 2013.
- [90] Soumya K Ghosh Monidipa Das. Data-Driven Approaches for Spatio-Temporal Analysis: A Survey of the State-of-the-Arts. *Journal of Computer Science and Technology*, 35(3):665, 2020.
- [91] Abdullah Mueen. Time Series Motif Discovery: Dimensions and Applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(2):152–159, 2014.
- [92] Daniel Mullner and others. Fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, 53(9):1–18, 2013.
- [93] Fionn Murtagh and Pedro Contreras. Algorithms for Hierarchical Clustering: An Overview. *WIREs Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.
- [94] Garima Nautiyal, Sandeep Maithani, Ashutosh Bhardwaj, and Archana Sharma. Entropy-Based Approach for the Analysis of Spatio-Temporal Urban Growth Dynamics. In *Multidisciplinary Digital Publishing Institute Proceedings*, page 17, 2019.
- [95] Met Office. Met Office Integrated Data Archive System (MIDAS) Land and Marine Surface Stations Data (1853-current). Technical report, 2012.
- [96] Stefanos Ougiaroglou and Georgios Evangelidis. Efficient Dataset Size Reduction by Finding Homogeneous Clusters. In *Fifth Balkan Conference in Informatics*, pages 168–173. ACM, 2012.
- [97] Bei Pan, Ugur Demiryurek, Farnoush Banaei-Kashani, and Cyrus Shahabi. Spatiotemporal Summarization of Traffic Data Streams. In *ACM SIGSPATIAL International Workshop on GeoStreaming*, pages 4–10. ACM, 2010.

- [98] Karl Pearson. LIII. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.
- [99] Thanh Noi Phan and Martin Kappas. Application of MODIS Land Surface Temperature Data: A Systematic Literature Review and Analysis. *Journal of Applied Remote Sensing*, 12(4):1–20, 2018.
- [100] Bernhard Preim and Kai Lawonn. A Survey of Visual Analytics for Public Health. *Computer Graphics Forum*, 39(1):543–580, 2020.
- [101] Alicia Quirós, Raquel Montes Diez, and Dani Gamerman. Bayesian Spatiotemporal Model of fMRI Data. *NeuroImage*, 49(1):442–456, 2010.
- [102] Peter Ranacher, Richard Brunauer, Wolfgang Trutschnig, Stefan Van der Spek, and Siegfried Reich. Why GPS Makes Distances Bigger Than They Are. *International Journal of Geographical Information Science*, 30(2): 316–333, 2016.
- [103] K R Rao and P Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Elsevier Science, 2014.
- [104] Andrew P Reimer and Elizabeth A Madigan. Veracity in Big Data: How Good is Good Enough. *Health Informatics Journal*, 25(4):1290–1298, 2019.
- [105] Filipe Rodrigues, Kristian Henrickson, and Francisco C Pereira. Multi-Output Gaussian Processes for Crowdsourced Traffic Data Imputation. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):594–603, 2018.
- [106] Sam T Roweis and Lawrence K Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.
- [107] Joelson Santos, Talat Syed, Murilo Coelho Naldi, Ricardo JGB Campello, and Jörg Sander. Hierarchical Density-Based Clustering using MapReduce. *IEEE Transactions on Big Data*, 2019.
- [108] F Sassite, M Addou, and F Barramou. A Smart Data Approach for Spatial Big Data Analytics. In *2020 IEEE International Conference of Moroccan Geomatics (Morgeo)*, pages 1–6, May 2020.
- [109] David Sathiaraj, Thana-on Punksam, Fahui Wang, and Dan P K Seedah. Data-driven Analysis on the Effects of Extreme Weather Elements on Traffic Volume in Atlanta, GA, USA. *Computers, Environment and Urban Systems*, 72:212–220, 2018.



- [110] Salar Shahini Shamsabadi, Yasamin Sadat Hashemi Tari, Ralf Birken, and Ming Wang. Deterioration Forecasting in Flexible Pavements Due to Floods and Snow Storms. In *EWSHM-7th European Workshop on Structural Health Monitoring*, 2014.
- [111] Razieh Sheikhpour, Mehdi Agha Sarram, Sajjad Gharaghani, and Mohammad Ali Zare Chahooki. A Survey on Semi-Supervised Feature Selection Methods. *Pattern Recognition*, 64:141–158, 2017.
- [112] Bin Shen and Tomonari Masada. Explaining Prices by Linking Data: A Pilot Study on Spatial Regression Analysis of Apartment Rents. In *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on*, pages 188–189. IEEE, 2014.
- [113] Amritpal Singh, Sahil Garg, Ravneet Kaur, Shalini Batra, Neeraj Kumar, and Albert Y Zomaya. Probabilistic Data Structures for Big Data Analytics: A Comprehensive Review. *Knowledge-Based Systems*, 188: 104987, 2020.
- [114] Priyanka Sinha. Multivariate Polynomial Regression in Data Mining: Methodology, Problems and Solutions. *International Journal of Scientific and Engineering Research*, 4(12):962–965, 2013.
- [115] R O Sinnott and W Voorsluys. A Scalable Cloud-based System for Data-Intensive Spatial Analysis. *International Journal on Software Tools for Technology Transfer*, 18(6):587–605, November 2016.
- [116] Sabina Sisovic, Marija Brkic Bakaric, and Maja Matetic. Reducing Data Stream Complexity by Applying Count-Min Algorithm and Discretization Procedure. In *4th International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 221–228, March 2018.
- [117] B Sivakumar, K-K Phoon, S-Y Liong, and C-Y Liaw. A Systematic Approach to Noise Reduction in Chaotic Hydrological Time Series. *Journal of Hydrology*, 219(3-4):103–135, 1999.
- [118] John Snow. *On the Mode of Communication of Cholera*. John Churchill, 1855.
- [119] Saúl Solorio-Fernández, J Ariel Carrasco-Ochoa, and José Fco Martínez-Trinidad. A Review of Unsupervised Feature Selection Methods. *Artificial Intelligence Review*, 53(2):907–948, 2020.
- [120] Liam Steadman. Reducing spatio-temporal data: Methods and analysis. <https://bitbucket.org/liamac/phd-thesis-final> (commit 1d9e0ee), 2020.

- [121] Liam Steadman, Nathan E Griffiths, Stephen A Jarvis, Stuart McRobbie, and Caroline Wallbank. 2D-STR: Reducing Spatio-temporal Traffic Data-sets by Partitioning and Modelling. In *5th International Conference on Geographical Information Systems Theory, Applications and Management*, pages 41–52, 2019.
- [122] Liam Steadman, Nathan Griffiths, Stephen Jarvis, Mark Bell, Shaun Helman, and Caroline Wallbank. Reducing and Linking Spatio-Temporal Datasets with kD-STR. In *3rd ACM SIGSPATIAL Workshop on Advances in Resilient and Intelligent Cities*. ACM, 2020.
- [123] Liam Steadman, Nathan Griffiths, Stephen Jarvis, Mark Bell, Shaun Helman, and Caroline Wallbank. kD-STR: A Method for Spatio-Temporal Data Reduction and Modelling. *ACM Transactions on Data Science*, 2(3), 2021.
- [124] He Su, Weidong Kang, Yuanjun Xu, and Jiading Wang. Assessing Groundwater Quality and Health Risks of Nitrogen Pollution in the Shenfu Mining Area of Shaanxi Province, Northwest China. *Exposure and Health*, 10(2):77–97, 2018.
- [125] Kai Sheng Tai, Vatsal Sharan, Peter Bailis, and Gregory Valiant. Sketching Linear Classifiers over Data Streams. In *International Conference on Management of Data*, pages 757–772. ACM, 2018.
- [126] Athanasios Theofilatos. Incorporating Real-Time Traffic and Weather Data to Explore Road Accident Likelihood and Severity in Urban Arterials. *Journal of Safety Research*, 61:9–21, 2017.
- [127] Sahar Torkamani and Volker Lohweg. Survey on Time Series Motif Discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2):e1199, 2017.
- [128] Roberto Trasarti, Fabio Pinelli, Mirco Nanni, and Fosca Giannotti. Mining Mobility User Profiles for Car Pooling. In *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1190–1198, New York, NY, USA, 2011. ACM.
- [129] Isaac Triguero, Joaquín Derrac, Salvador Garcia, and Francisco Herrera. A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):86–100, January 2012.
- [130] United Nations. World Urbanization Prospects (The 2018 Revision). Technical report, August 2019.

- [131] Mingliang Wang, Han-Xiong Li, and Wenjing Shen. Deep Auto-Encoder in Model Reduction of Large-Scale Spatiotemporal Dynamics. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3180–3186, July 2016.
- [132] S Wang, J Cao, and P Yu. Deep Learning for Spatio-Temporal Data Mining: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.
- [133] Zhangang Wang, Honggang Qu, Zixing Wu, Hongjun Yang, and Qunle Du. Formal Representation of 3D Structural Geological Models. *Computers & Geosciences*, 90:10–23, 2016.
- [134] Joe H Ward Jr. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.
- [135] M Whelan, N A L Khac, and M Kechadi. Data Reduction in Very Large Spatio-Temporal Datasets. In *19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pages 104–109, 2010.
- [136] Michael Whelan, Nhien-An A Le-Khac, and M-Tahar Kechadi. Comparing Two Density-Based Clustering Methods for Reducing Very Large Spatio-Temporal Dataset. In *IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*, pages 519–524, June 2011.
- [137] Liang Ze Wong, Huiling Chen, Shaowei Lin, and Daniel Chongli Chen. Imputing Missing Values in Sensor Networks Using Sparse Data Representations. In *17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 227–230. ACM, 2014.
- [138] Alan Woodley, Ling-Xiang Tang, Shlomo Geva, Richi Nayak, and Timothy Chappell. Using Parallel Hierarchical Clustering to Address Spatial Big Data Challenges. In *IEEE International Conference on Big Data (Big Data)*, pages 2692–2698. IEEE, 2016.
- [139] Chunrui Wu, Junfeng Tian, and others. Spatio-Temporal Outlier Detection: A Survey of Methods. *International Journal of Frontiers in Engineering Technology*, 2(1), 2020.
- [140] K Wu, D Lee, A Sim, and J Choi. Statistical Data Reduction for Streaming Data. In *2017 New York Scientific Data Summit (NYSDS)*, pages 1–6, August 2017.

- [141] Y Xiong, D Bingham, W J Braun, and X J Hu. Moran’s I Statistic-Based Nonparametric Test with Spatio-Temporal Observations. *Journal of Nonparametric Statistics*, 31(1):244–267, 2019.
- [142] Chi Yang and Jinjun Chen. Chapter 4 - Efficient Nonlinear Regression-Based Compression of Big Sensing Data on Cloud. In Hui-Huang Hsu, Chuan-Yu Chang, and Ching-Hsien Hsu, editors, *Big Data Analytics for Sensor-Network Collected Intelligence*, pages 83–98. Academic Press, 2017.
- [143] Xiaojun Yang, Weizhong Yu, Rong Wang, Guohao Zhang, and Feiping Nie. Fast Spectral Clustering Learning with Hierarchical Bipartite Graph for Large-Scale Data. *Pattern recognition letters*, 2018.
- [144] Ilya Zaliapin, Andrei Gabrielov, Vladimir Keilis-Borok, and Henry Wong. Clustering Analysis of Seismicity and Aftershock Identification. *Physical Review Letters*, 101(1):018501, July 2008.
- [145] Huichu Zhang, Yu Zheng, and Yong Yu. Detecting Urban Anomalies Using Multiple Spatio-Temporal Data Sources. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):54:1–54:18, 2018.
- [146] Wei Zhang, Gongxuan Zhang, Xiaohui Chen, Yueqi Liu, Xiumin Zhou, and Junlong Zhou. DHC: A Distributed Hierarchical Clustering Algorithm for Large Datasets. *Journal of Circuits, Systems and Computers*, 28(04):1950065, 2019.
- [147] Yang Zhang, Rong Wang, Shouzhe Li, and Shengbo Qi. Temperature Sensor Denoising Algorithm Based on Curve Fitting and Compound Kalman Filtering. *Sensors*, 20(7):1959, 2020.
- [148] Yuan Zhao and Xinchang Zhang. Calculating Spatial Configurational Entropy of a Landscape Mosaic Based on the Wasserstein Metric. *Landscape Ecology*, 34(8):1849–1858, 2019.
- [149] Guanghu Zhu, Jianpeng Xiao, Tao Liu, Bing Zhang, Yuantao Hao, and Wenjun Ma. Spatiotemporal Analysis of the Dengue Outbreak in Guangdong Province, China. *BMC Infectious Diseases*, 19(1):493, 2019.